STUDIES BY COMPUTER SIMULATION OF VISUAL DETECTION AND DISCRIMINATION OF IMAGES OF ASTRONOMICAL OBJECTS

Colin Perry
Physics Department,
Imperial College (London)

Thesis for degree of PhD.

1985

ABSTRACT

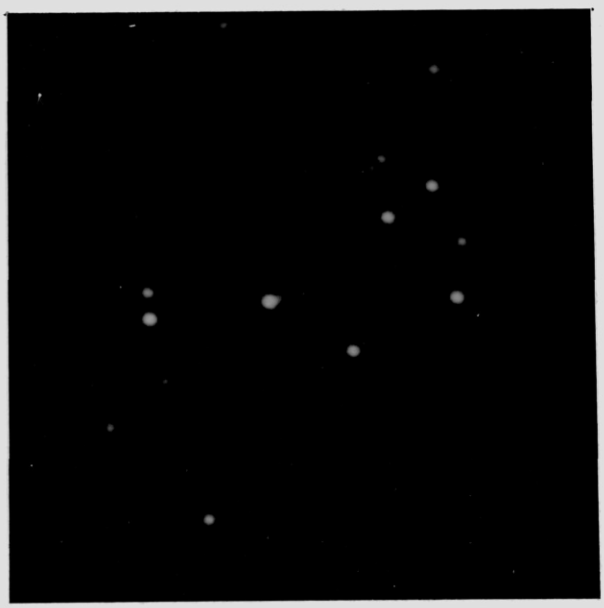
A simple but effective image processing system is described, implemented on a general purpose scientific computer (CDC 174) with a standard microfilm plotter for output, producing both contour maps and continuous tone images. It handles images up to 220 pixels square, easily increased to 512. Two filters are included, one capable of performing deconvolution. Full Fortran listings are given for this system and other programs used.

Also described are routines to synthesise images, with fields of stars or elliptical galaxies and pseudorandom noise, and to perform similar synthesis and manipulation on one dimensional (spectral) data. The user interface is provided by a general purpose command interpreter, and a generator program builds the system for a particular application.

The detection of peaks in spectra, lines in images of spectra, and images of stars, was studied, and results are presented showing general agreement with theory, indicating the near optimal performance of visual assessment of images, and demonstrating some practical requirements for satisfactory results. The positional accuracy possible is also demonstrated. Resolution was studied: it is found slightly better than indicated by the usual criteria of resolution, and the degradation produced by noise is described. Doubleness of stars is shown visually detectable to separations half or less those needed for resolution.

Sharpening by partial deconvolution is demonstrated in both one and two dimensions and its properties and limitations described. Sharpening by a factor of 1.7 in one dimension or 1.5 in two are suggested as the most ever usable. It is shown to be of little general utility, but suitable in special cases.

Elliptical galaxies are shown to be most detectable as diffuse objects at a moderate size: the drop in sensitivity away from this is given. The reduction in apparent ellipticity of small and faint galaxies is described.



CONTENTS

	Abstract	2
	Contents	4
	Illustrations	5
	Preface	7
	1: Introduction	
1.1	Aims and Methods	9
	2: Computing	
2.1	Computing: Overview	18
2.2	Plotting System	23
2.3	Image Generation System	33
2.4	One Dimensional System	37
2.5	Command Interpreter and Generator Program	40
2.6	Computing: Assessment	44
	3: Results	
3.1	Peaks in One Dimension	51
3.2	Peaks in Two Dimensions	59
3.3	Resolution in One Dimension	67
3.4	Resolution in Two Dimensions	76
3.5	Deconvolution in One Dimension	90
3.6	Deconvolution in Two Dimensions	103
3.7	Elliptical Galaxies	109
	4: Conclusion	
4.1	Summary	127
4.2	References	130
	A: Programs	
A1	Plotting	132
A2	Image Generation	157
A3	One Dimensional	176
A4	Command Interpreter and Generator	186
	B: Sample Input	
B1	To Plot a Star Field	202
B2	For Generator Program	203
	C: *PLATES*	206
	Select Bibliography	217

LIST OF ILLUSTRATIONS

Figures

1.1a	Double gaussian profiles	16
2.2a	Method of plotting continuous tones	30
2.2b	Sample frame with continuous tone plot	30
2.2c	An alternative method of plotting continuous tones	31
2.2d	Sample frame with contour plot	31
2.2e	Sample line printer output	32
3.1a	One dimensional peaks in noise	57
3.1b	Variation in appearance of weak lines	58
3.2a	Contour plots of stars before and after matched filtering	64
3.2b	Comparison of linear and logarithmic spacing of contours	65
3.2c	Contour plots of stars	66
3.3a	Resolution in one dimension: double lines	70
3.3b	Appearance of double lines	71
3.4a	Resolution in two dimensions: double lines	82
3.4b	Contour plots of double stars	83
3.4c	Sample frame for testing sensitivity to non-circular contour	88
3.4d	Visual sensitivity to distortion of contour of double stars	89
3.5a	One dimensional deconvolution filters	97
3.5b	Example of deconvolution in one dimension	98
3.5c	Signal to noise degradation in one dimensional deconvolution	99
3.5d	Effect of filter length on one dimensional deconvolution	100
3.5e	Effect of error in assumed width on one dimensional deconvolution	100
3.5f	Effect of distortion on one dimensional deconvolution	101
3.5g	Effect of aliasing on one dimensional deconvolution	101
3.5h	Effect of sampling on one dimensional deconvolution	102
3.6a	Ideal profiles produced by two dimensional deconvolution	107
3.6b	Two dimensional deconvolution filters	107
3.6c	Signal to noise degradation in two dimensional deconvolution	108
3.7a	Theoretical detection sensitivity for elliptical galaxies	116
3.7b	Actual detection sensitivity for elliptical galaxies	117
3.7c	Elliptical galaxies in contour plots	118
3.7d	Elliptical galaxies in contour plots	121
3.7e	Apparent shape of elliptical galaxies	125

Tables

1.1	Separation to resolve a pair of gaussian profiles	15
3.7	Integrated flux as a function of radius for a deVaucouleurs profile	115

Plates

-	The Pleiades plotted from star catalog, image 200 pixels square. To magnitude 8.	3
2.2a	Effect of jitter on continuous tone plotting	207
3.1a	One dimensional peaks in noise	207
3.1b	Line depth needed for visibility	207
3.1c	Emission lines in noise	208
3.1d	Absorption lines with noise	208
3.2a	Star series in noise	208
3.2b	Stars in noise	209
3.2c	Suppressing a noisy background	209
3.2d	Disadvantages of suppressing a noisy background	209
3.3a	Appearance of double lines	210
3.4a	Appearance of double stars	211
3.5a	Example of deconvolution in one dimension	212
3.6a	Example of deconvolution in two dimensions	212
3.7a	Elliptical galaxies	213
3.7b	Elliptical galaxies	214

PREFACE

The work here described was done between 1981 and 1983 in the Astronomy Group of the Physics Department at Imperial College.

The college computer centre provided a most reliable service, and the staff were uniformly cooperative. Their help with the phototypesetter is particularly appreciated.

I am grateful to Professor J. Ring and Dr. B. L. Morgan for their forbearance and the continued provision of the computing resources needed. My thanks also go to my parents, who supported me during the period of this work. A previous SRC studentship is acknowledged.

C. Perry,
July 1983.

1.1 Aims and Methods

1.1.1 Objectives

The problems to which this work was addressed were some of those concerning the evalulation of astronomical images, and, as a secondary concern, spectra, whether recorded as images or otherwise. In particular the effectiveness of direct visual assessment was determined and where possible compared with theoretical limits or the results after various types of processing. The work was mostly confined to the simplest of subjects, single and double star images and single and double spectral lines, both in the presence of noise. It was extended to cover also the images of elliptical galaxies.

The general aim was to characterise the effectiveness of traditional visual methods, and of the more basic of modern methods using numeric processing of data scanned either directly or from a photographic image. The comparison would give an indication of what, if any, gains would be expected from the newer, more expensive, techniques, and indicate how in suitable cases those techniques might be best applied. In other cases the absolute rather than comparative effectiveness was of interest, especially in the resolution of double stars and lines. Systematic errors appear in the visual classification of all but the simplest of images, and their potential magnitude needs to be determined: elliptical galaxies are one such case.

A subsiduary aim, given the basic method selected, was to demonstrate the feasibility of conducting a useful range of image processing on a general purpose computer system, and to evaluate performance and costs. It was not intended that the programs used should form a fully refined, fully documented, image processing system suitable for general use, but they were designed to allow development into such.

1.1.2 Choice of Method

Given the general aims of the investigation, a number of methods were considered. Each is particularly suited to certain types of problem, and to that extent some of them are complementary, but to develop more than one would have been prohibitively time-consuming.

It would have been possible to base the work on real rather than synthetic data. This was not seriously considered, not being practical without ready access to both an extensive library of original astronomical plates and a suitable scanning method.

Purely optical methods are possible, the most practicable using superimposed photographic transparencies. Separate image components, such as stars, galaxies, and a noisy background, can be printed on film to various densities and at different scales, and then superimposed in different spatial relationships. The main difficulty is in keeping tight control over the photographic process. There is also the problem of making the initial images, and while optical methods are possible computer generation is more reasonable. The method is potentially very suitable for investigating visual search efficiency for small galaxies in extensive star fields, the appearance of chance alignments of galaxies, and similar problems. It was rejected as calling for better photographic facilities than were available and being too specialised in application.

The obvious method would be to use a dedicated image processing computer system with a graphics terminal capable of showing continuous tone images. Its truly interactive nature makes it especially suited to establishing thresholds for detection in noise and for resolution, but if provided with a good hardcopy device it is of quite general application. Unfortunately the display capability required is well beyond that of any normal microcomputer, and so an existing system would have to have been used. These are expensive and still not common. Only one might possibly have been available, that at University College, University of London, forming part of the Starlink network. It was not clear that authorisation to use it could be obtained, and other disadvantages led to its rejection. Location at a different site would have made it inconvenient, especially for writing and testing the image generating programs that would still have been required: there is no adequate remote access from IC. Conversations with users also revealed that the system, especially the graphics facility, was heavily used during the day and there would often be no free terminal: evening working was very restricted. It also indicated considerable dissatisfaction with much of the software, some of the most basic required functions being either not present or unusable. Apparently at least four users have written their own contouring programs, the official version being unsatisfactory. The system also lacks any proper hardcopy capability: users have to resort to photographing a graphics diplay with improvised equipment.

Computer generated images can also be used without a real-time display, relying on a hardcopy device, at least for continuous tone images. This is well within the capability of a microcomputer: it would be slow, but not intolerably so. A high quality hardcopy device, capable of at least 512 by 512 pixel resolution and with much better quality than direct screen photography, could be built from a normal rather than continuous tone monitor display using a camera mechanically scanned vertically, the price again being low speed, perhaps 3 minutes to plot a frame. Serious consideration was given to constructing such a system and it was rejected mainly because the uncertainty in the time needed to develop the hardcopy device, added to that in writing the image processing software, was too great.

The alternative way to get computer generated images is to use a normal, general purpose, computer system, and this was the method selected. The Imperial College system had no device specifically for producing continuous tone images, but initial tests demonstrated that satisfactory images could be produced on microfilm. The advantage of using the college system, besides convenience, was

that it was being under-used and this work was done with resources that would otherwise have been waste. In a more local sense, it was done within a surplus of the resources allocated to the Astronomy Group and paid for even if not used.

1.1.3 Conduct of Tests

Much of the work was concerned with visual discrimination rather than objective measurements, and the way in which tests of this sort were conducted needs some justification, for no other observer than the author was used. This was a deliberate decision made as a matter of expediency: the reasons for it, and the probable consequences, will be discussed.

It was felt that to gain any meaningful statistical results, a substantial number of test subjects would be required. Certainly more than 2 or 3: little can be deduced from such a small sample of observers, whether their performances agree or disagree. Probably 5 or 6 would be about the sensible minimum.

Administering numerous tests to this number of people would call for a type and degree of organisation more familiar in experimental psychology labs than astronomy. To do so efficiently would have needed each test to be put in a suitable form. With some it would have meant the preparation of a properly presented series of test images. Others, that were done interactively at the terminal, would have had to be greatly speeded up and simplified, and means for recording the results added. It is difficult to say how much extra work all this would entail: a three fold increase seems conservative.

There would also be considerable difficulty in obtaining the subjects. Many of the tests were lengthy to evaluate, a particular series of images often needing a couple of hours, and if done with useful care were a considerable mental strain. It was not reasonable to expect members of the Astronomy Group, however willing, to undergo such tests. Even if they did, it would be difficult to motivate them highly so as to give their best performance, as good as they would give if it was their own research data they were examining. The usual psychology lab approach of calling for volunteers, often from the general student body, and paying them at a fairly nominal rate for their time, was impractical: there were not the needed funds, and the subjects would lack both motivation and the prior experience that helps make a good observer.

Thus it appeared that to use a significant number of subjects for the tests would mean greatly reducing the number of tests, and it was felt that a relatively comprehensive coverage was desirable before more rigorous tests limited to a few aspects. Repetition of a few selected tests on a number of subjects, preferably practiced astronomical observers, would now be justified.

So all the tests were evaluated by the one observer. From prior experience, astronomical and otherwise, the author can claim to have acute and reasonably practised eyesight. Certainly he was adequately motivated.

The majority of tests would not be expected to be much affected by the observer's ability, as they involved discrimination in the presence of sufficient noise to greatly impair performance. In these, individual cases were usually quite clear cut, and statistical noise from the inevitable smallness of the samples was undoubtedly the main uncertainty in the results. A second, smaller category of tests while certainly affected by the observer's ability would not be expected to be greatly so in fractional terms because the appearance alters so rapidly with the

parameter being determined. The resolution of double lines in the absence of noise is of this type. Finally a very few tests depend solely on the observer's ability and their results will be entirely dependent on it. Such a case is the test of the ability to discriminate a slight ellipticity of contour from true circularity. In those tests sensitive to the observer's powers, and especially those in the last category, while the absolute values obtained may not be representative the form of the results with varying parameters should be reliable.

A consequence of not using other subjects was that the observer's honesty could be assumed. Thus it was not in all cases necessary to completely randomise the test patterns. For example, a pattern to test for detection of stars in noise could have the stars on a regular grid, and the detectability of each judged in terms of the noise over the entire frame and disregarding its known position. This could save a great deal of time, but was done with discretion. Often partial randomisation was judged sufficient: so a double star test pattern might have the separation regularly varying over a grid of double stars, with only the position angle randomised. When either necessary or convenient, test patterns were fully randomised, as for example in the tests for sensitivity to ellipticity of contour.

1.1.4 Profiles Used

Three types of astronomical image components are considered in this work: spectral lines, stars, and elliptical galaxies. None of them have a simple theoretically determined form and their observed shapes vary considerably. It was necessary to choose in each case a single profile that was adequately representative and as simple as possible: to have used either collections of experimental profiles or of the models that have been proposed to fit them would have been prohibitively tedious with very little probable gain. With only a few exceptions, which will be noted, the exact form of the profile should make little difference.

Spectral lines show the widest variety of shape. Their natural Lorentz profiles are modified by saturation, and by doppler broadening. Closely spaced spectral components are blended, lines from regions of the source with different conditions are combined, and so are lines from separate sources with different doppler shifts due to radial velocity. The observed profiles are further modified by the finite slit width of a spectrograph, its diffraction limited resolution as modified by optical aberrations, and the spread function of the photographic plate or other detector. Sometimes a single source of line width can be dominant and result in a near gaussian profile: this can be the case with radial velocity broadening of the integrated light from a globular star cluster. More generally many factors will be modifying the line profile, and most will contribute a quite non-gaussian shape. But the blending of them will tend to weaken their individual characteristics, and the line profile can be expected to tend towards the gaussian.

Thus the gaussian profile was chosen to represent spectral lines: a simple single parameter shape, and analytically tractable. It has two limitations. The natural line profile has extensive wings, and these will often be little affected by other sources of width. Any result depending on the shape of the profile at a level below about 5% of the peak must be applied to real spectral lines with caution. This only affects very unequal double lines in this work, and because the slope of the natural profile falls in the wings much faster than the profile itself, the errors introduced are

essentially negligible. The second limitation is that if the true profile has a fourier transform that falls to zero at a finite spatial frequency, the possible extent of sharpening by deconvolution is restricted. Because the degree of sharpening is limited for other reasons, this is only significant if most of the observed line width is due to a broad slit. In practice, in any case that even moderate resolution was desired, the spectrograph slit would be narrowed till other factors were contributing at least as much to the line width as the slit, and then the zero introduced into the spatial frequency spectrum would be at a high enough frequency to not limit deconvolutional sharpening. The process is not otherwise much affected by the line profile.

A star image is just the time-averaged point spread function of the instrument and the atmosphere. The case of interest is when the telescope is large, so its diffraction pattern is negligible, and its imperfections also negligible, and the limited resolution of the photographic plate has only a small effect. The size and shape of the star image is then determined almost entirely by the atmosphere. As might be expected of a random process, the image is circularly symmetric and approximately gaussian in profile. Experimentally determined profiles are not exactly gaussian. Often they are fitted by a similar profile but with the power of the radial distance r appearing in the exponent slightly altered, or by a gaussian with a weak exponential component added. There is no generally accepted result, and there are a number of factors that would be expected to cause the profile to vary somewhat. For example, scattering in the air, telescope, or plate could add an exponential halo. Variation of the width of the point spread function during the time of the exposure could distort the gaussian profile, typically making the power of r appear less. Guiding and focus errors could have the opposite effect, and so could 'edge effects' in the photographic process.

Overall, a gaussian profile seems as good a representation of a typical star image as any and was selected for this work. It had certain advantages besides simplicity and having only a single parameter. Using the same profile as for spectral lines enabled some of the results to be related more readily, and some of the calculations to be shared. It had computational advantages in that a matched filter could be implemented efficiently as two two orthogonal one dimensional filters. The only limitation is that real images may differ slightly in the extent of their 'wings', and so the resolution of very unequal double stars will be slightly affected. There is another limitation inherent in the general form of representing the image: the instantaneous image, especially when seen in a restricted waveband, shows a random diffraction limited speckle pattern. This is averaged out over long exposures, being reduced by a factor of about 100 in 10 minutes, but might still set a theoretical limit to the possible extent of deconvolutional sharpening. (The practical limit is much less).

An elliptical galaxy appears closely elliptical with a radial profile of constant form but varying in scale according to direction. The slight twist in orientation that may be observed between the apparent principal axes of the inner and outer parts is negligible for the purposes of this work. The radial profile, with the effects of the system point spread function removed, varies between galaxies. There is no theoretical ideal profile, so a number of empirical models have been fitted. Chief of these are the Hubble, King, and de Vaucouleurs profiles. The Hubble profile goes as $1/(1+(R/Ro))^2$. It is physically unreasonable in that the total flux integrates to

infinity at infinite radius, and for the same reason inconvenient to use in this work. The King profile is borrowed from a physical model of globular clusters in which tidal effects cause a cutoff at large radius. The model is not valid for galaxies, and the profile is inconvenient to use because it is charaterised by two parameters, a core and a cutoff radius. The de Vaucouleurs profile has a single parameter, the effective radius Re, is easily calculated, and gives a finite total flux. It is sometimes known as the 'r to one quarter' law. The variations between galaxies and the uncertainties of the experimental data are such that no one profile, of these or others, is clearly superior. The de Vaucouleurs profile is as good as any, is the most widely used, and is much the most convenient for this work: it was thus the one chosen.

1.1.5 Resolution Criteria

The one theoretical criterion of resolution that is widely used is Rayleigh's. This applies to the diffraction limited images formed by an optical system with a rectangular aperture, as in most spectrographs, or circular, as in telescopes. It defines the image of a pair of equally bright point objects (or lines, for a rectangular aperture) as being resolved when their separation corresponds to the distance from the centre of the individual diffraction pattern to the first zero. At this separation the image shows two equal intensity maxima with a dip in between. This dip is by 26.5% of the maximum for a circular aperture, 18.9% for a rectangular.

When the image is not one of these forms, the Rayleigh criterion is not directly applicable, but it is usually taken that requiring a dip in intensity of the same depth will be an equivalent criterion. Often the required dip is rounded off to 20%. This criterion can be applied when the two components of the image are not equally bright, the dip then being relative to the maximum corresponding to the weaker component.

However, Rayleigh defined his criterion as corresponding to resolution by casual observation, and practice shows it to be conservative. The best established empirical result is Dawes limit for the resolving power of a telescope. Dawes, a double star observer, published this in 1867 (Rayleigh's criterion was only published in 1879). It states that the resolving power in seconds of arc is 4.56 divided by the aperture in inches. It was generally accepted, with some observers claiming slightly, perhaps 10 or 15%, better resolution. This limit corresponds very closely to a diffraction limited image with zero dip, that is, to an intensity profile with a flat top. Less precisely, the resolving power of the optical microscope has been found in practice to be well given by the early result of Abbe, even though this was derived for a periodic object, and when the effects of partially coherent illumination are allowed for in the light of the methods of practical microscopists, this too is found to correspond to zero dip and a flat topped image. Practical spectroscopists (such as Sawyer, 1944) recognise the presence of any dip in a line profile as indicating resolution.

Some optical designers use the same criterion, either in its zero dip form, sometimes known as the Sparrow criterion, or as a resolving power for a diffraction limited system that is its equivalent. One such was Conrady (1929), and he also gives the apparent visual diameter of the central maximum of the diffraction pattern for a circular aperture, based on numerous careful tests, as the same as his

value for the resolving power. This provides some slight justification for the astronomical practice of quoting an apparent diameter of a star image seen through a large telescope as 'the seeing', as if it was the same as the limit of resolution.

Other criteria have sometimes been used. One, the Taylor criterion, is of attractive simplicity: it is that the resolving power is just the full width at half height of the system spread function. This is a measure more used in particle physics. In optical systems it should be reserved for very rough work, as while it tends to be numerically similar to other measures of resolution it is unrealistically sensitive to 'wings' or 'overshoot' in the spread function.

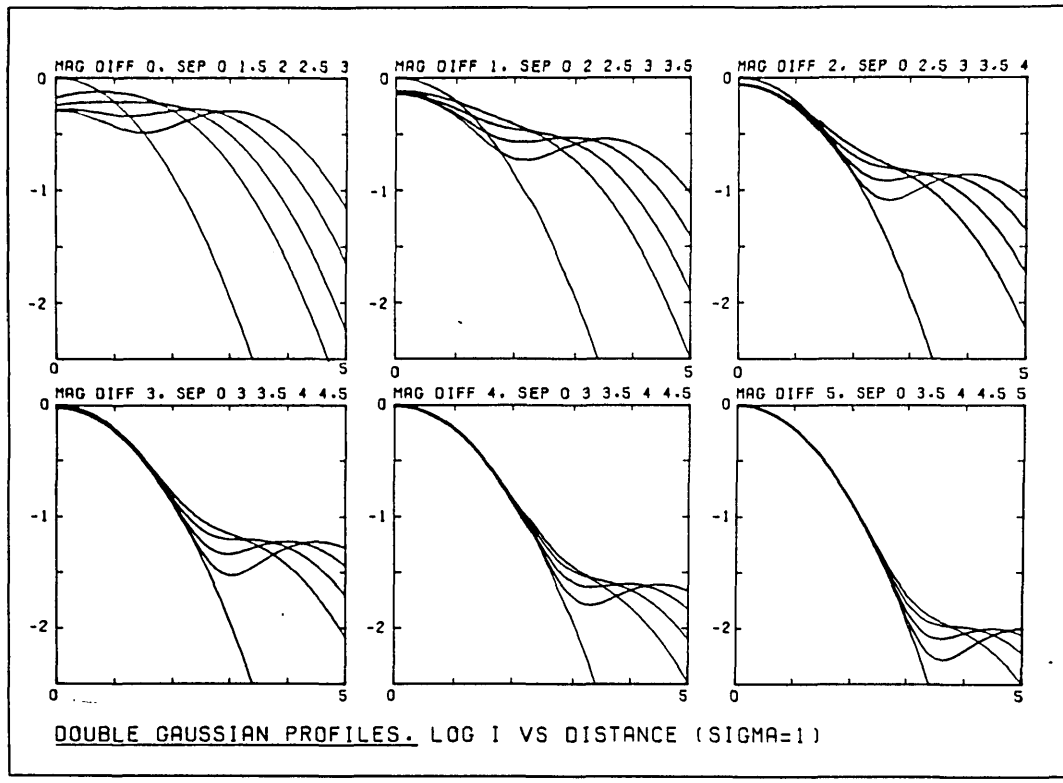
It appears that the zero dip criterion is a better approach to practice than the Rayleigh criterion or the similar 20% dip. While not as easy to assess as the width at half height it is better behaved and can be applied to the resolution of unequal components. It was therefore chosen for use when a criterion of resolution was required.

For the diffraction limited image of a pair of equally bright point objects the zero dip criterion corresponds to a separation of 78% of Rayleigh's, and the full width, half height to 84%. For a gaussian profile image, zero dip gives a separation 75% of the 20% dip criterion (approximately equivalent to Rayleigh's), and the full width, half height to 89%.

Details of the separation required between gaussian profiles for various depth dips are given in Table 1.1. Note that the ratios of intensities are given in astronomical magnitudes, that is as 2.5log(ratio). For equal components zero dip is at exactly 2 sigma separation. For unequal components it can be approximated by $2+(\text{mag.diff.})^{0.5}$ in units of sigma, accurate to within 10% up to 40 magnitudes. Over all this range a 20% dip needs a separation greater by about ½ sigma, and 50% by about 1 sigma. The variations of the combined profile with separation and magnitude difference are illustrated in Fig. 1.1a.

Table 1.1 Separation to resolve a pair of gaussian profiles.
Accuracy: possible error 1½ in last place.

Magnitude Difference	Dip Percent					
	0	1	5	10	20	50
0	2.00	2.11	2.27	2.41	2.65	3.32
0.5	2.48	2.54	2.65	2.76	2.97	3.58
1	2.75	2.81	2.92	3.02	3.22	3.80
2	3.17	3.23	3.33	3.44	3.63	4.18
5	4.09	4.14	4.25	4.35	4.53	5.05
10	5.18	5.24	5.36	5.46	5.64	6.15
20	6.82	6.88	6.99	7.10	7.28	7.79
40	9.19	9.26	9.38	9.50	9.68	10.18



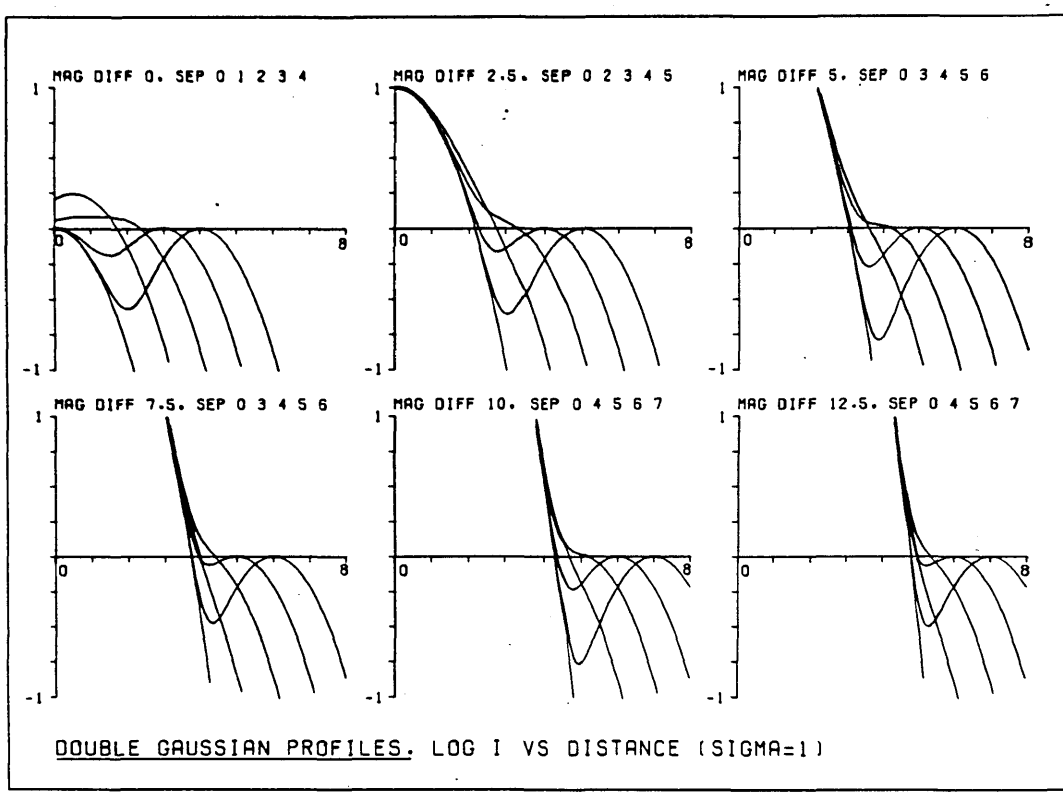


Fig.1.1a Double gaussian profiles.

2.1 Computing: Overview

2.1.1 Requirements

The capability most critical to this work, and the one implemented first, was the production of continuous tone images on the microfilm plotter. This is a role for which the device was not designed, and to which it is not well suited. The image size in pixels that would have to be handled could not be exactly defined at the start. As a guide it was noted that image processing systems handling space probe pictures commonly work on 512 by 512 images. Since this work would not be concerned with fine image detail, a rather smaller size, 200 by 200, was felt to be in general adequate, provided some ability to handle larger images was retained. In astronomical terms, this would adequately represent a field 1.5 arcmins across with very good (one arcsec) seeing. In visual terms, very good quality half-tone printing is possible with a 120 lines per inch screen, at which a 200 by 200 pixel image would be 40mm square. Because the cost of processing an image is roughly proportional to the number of pixels in it, the smallest possible size was generally used, and for most of this work images were between 60 and 120 pixels on a side.

A second type of image presentation was called for: contour mapping (that is, isophotes). This was needed primarily for comparison with continuous tone images, but also proved very useful for examining synthesised images on the graphics terminal. Because of its secondary role it did not need to be of the highest quality, and slightly angular contours and occasional artifacts could be tolerated, allowing a relatively simple adaptation of the continuous tone routines. It was efficient enough that a third form of image output, using a line printer to give a rough preview of the image, though implemented was never used.

The basic type of image to be generated was a field of one or more 'stars', that is, of peaks with circular symmetry and a given radial profile. Each star was specified in a table. Means to set up the table were needed, and these were made general. A regular grid might be required, or a random field. A parameter might vary in a regular fashion, form a regular series but be shuffled into a random order, or be truly random. To generate fields of double stars, the table could be set up with descriptions of each pair, and then converted into a table of individual stars. A field might contain a large number of stars, and so the process had to be efficient: since an image might be too large to be held in memory, it had to be generated a

line at a time. It was a simple extension to include the generation of elliptical galaxy images; the process was the same, with a different radial profile and extra parameters in the table specifying the distortion into an elliptical shape.

To a generated image noise could be added. An image could be filtered, either with a gaussian filter or a square aperture, or with an externally specified filter. Images could be summed, their statistics measured, and peaks located. This rather limited set of manipulations proved sufficient.

To complement these two dimensional capabilities, much simpler means for processing one dimensional data were provided. Synthesis and processing were analogous, while presentation was either as simple graphs or by conversion into two dimensional images resembling photographic spectra. A simple extension enabled it to handle two dimensional circularly symmetric data, specifically to construct and evaluate two dimensional deconvolution filters. Other additions allowed it to be used to prepare graphical illustrations.

2.1.2 Resources and Language

The computer used for this work was a CDC Cyber 174, the more powerful of the pair in the Imperial College computer centre (the other being a 170/720 with about half the power of the 174). This machine is a medium size dual processor scientific mainframe, roughly 5 times the speed of a VAX11/750. It is now somewhat dated in having only 131,000 words of memory (982 kbytes). These machines together support about 300 terminals and share about 4000 Mbytes of disc storage.

About 30 of the terminals are Tektronix storage tube graphics terminals, and it was one of these, a 4010, that was used most of the time. Also available are large screen 4014 terminals, linked to an electrostatic plotter (useful for previewing output, though not giving high quality). A high speed (40,800 baud) data link to the University of London computer centre enables output to be sent to the Calcomp 1670 microfilm plotter at that site: the output is returned by courier with a 1 or 2 day turnround. The 1670 was installed in 1972 but is still a relatively high performance device, producing line graphics at high speed (typically several thousand line segments a second) and with high resolution (it has a positional resolution of up to 16,000 on each axis). Output on 35mm film was used, with transparent lines on a black background.

The system has about 3000 users, and during the afternoon there are usually about 200 users logged in to the time-sharing service. There are also several thousand batch jobs run each week. Consequently the resources available to each user are limited (and the response can be exasperatingly slow). The situation improves in the evenings with only about 50 users, and the limit to the amount of memory that a program can use is increased, though only to 51,200 words, or 384 kbytes. All major programs were run then, and assume that much memory to be available. There were the subsiduary advantages that the cost of computer time is then half the daytime rate, and that the response is much faster. Mostly a substantial piece of work would be done in an elapsed time of between 2 and 5 times the central processor time used. Batch processing, which was not used, is in general subject to the same limit on available memory. Another consequence of the number of users is a considerable reluctance to allocate to a user more than 450 kbytes of disc space in which to retain files.

Little use was made of standard applications software except for the basic graphics packages. These were low level, providing little more than simple routines to plot lines, points or text, either displaying directly on the screen or producing a plot file that could be displayed or sent to the microfilm plotter. They provided very limited facilities for interactive use: essentially only the ability to read the position of a crosswire cursor set by thumb wheels.

The language used was Fortran. There was no choice, this being the only one for which graphics routines were available. The version used was an extended Fortran IV (the MNF5 compiler), rather than the more modern Fortran 77. This was a possibly overly conservative decision based on a preference for a compiler known to be trouble free: the F77 compiler did in fact have minor bugs that had to be fixed in the period of this work. The Fortran used was very close to F77 and lacked few of its functions that would have been useful. The disadvantage of using a non-standard Fortran, and consequent lack of portability of the programs, was considered acceptable in view of the extent to which the entire program structure had to be tailored to the particular type of machine and its dependence on a rather uncommon peripheral, the 1670.

2.1.3 Organisation

The basic capabilities required from the programs were to generate synthetic images, and then to plot them. This logical separation of function was reflected by the use of separate programs for generation and plotting. A single integrated program would have been preferable but would have had excessive memory requirements. The computer not having virtual memory, this would only have been possible by using overlays, a now essentially obsolete technique that was always awkward. The potential for trouble seemed to outweigh the slight benefits.

Images are transferred from the generation to the plotting programs as files. Within each program they are also stored as files, the shortage of available memory preventing their storage as arrays. In both cases the files are simple binary files of real numbers, each line of the image forming one record, and with no header or other extra records. This is entirely satisfactory for internal use, but somewhat inconvenient for communication between programs as the image size has to be supplied separately. The lack of a descriptive header, and the inefficiency of storing 60 bits per pixel, would also be disadvantages if the files were to be saved. Better file formats were devised, but the disadvantages of the very simple system initially operational proved so slight in the work for which it was used that they were left unimplemented. Similarly the use of random access files in place of sequential would have increased complexity to little benefit. The programs were written in such a way that both changes could have been made easily had they proved necessary.

A third program was produced, substantially simpler, to manipulate one dimensional data. For this no attempt at generality was made: all linear arrays of data share a preset length. All data are kept in a single array in memory. A very limited two dimensional capability was incorporated, the data using the same array. It is capable of generating image files acceptable to the two main programs.

Each of these programs is essentially a set of subroutines that perform different operations, and which the user wants to execute with parameters that he

specifies. The user does not want to have to write a new program to perform each individual job by calling the appropriate sequence of subroutines: a fixed program should read commands and parameter values from the terminal or from a file and for each execute the appropriate subroutine. To do this each of the three programs contains as well as its unique subroutines a command interpreter that is the same for each, except for the commands recognised and the subroutines called. Because it is shared by all three applications programs it was worth making relatively elaborate: an early simple version proved so effective that it was completely rewritten and fully repaid the effort. It is the primary user interface to the system, and ensures that all three programs look very similar to the user.

While the structure of the command interpreter is the same in all cases, and so is most of the code, the commands recognised and the subroutine calls generated are not. Writing these parts by hand would be tedious and error prone, and with every addition or alteration to the program its command interpreter would have to be modified. To avoid this a generator program was written that when given a file listing the subroutine names and the type of parameters required by each writes the corresponding part of the command interpreter to a file, using each subroutine name as a command. When the fixed part of the command interpreter is copied onto the end of it the interpreter is complete. The generator program also produces a procedure file to do this, and then add on all the files listed in its input as containing the source code for the required subroutines, and finally compile the whole program.

This allows the applications programs to be stored as many small files of subroutine source code, with for each program a file giving the script to be used as the input to the generator. This keeps the program in pieces of manageable size, easily replaced by new versions and of a reasonable length to print. Compiled code is not kept, which saves disc space and avoids problems of keeping compiled code updated when source code is changed. The computer time wasted in recompiling has not been significant, usually amounting to 10% or less of the total time used in a session at the terminal.

2.1.4 Status

Program design was a compromise between performance and ease of programming. Because it was intended for a specific, limited, use considerably lower efficiency and a certain lack of polish were acceptable that would not have been in a system for more general service. On the other hand flexiblity was essential as requirements could not be fully specified at the start but gradually evolved, and the system was used enough that it had to be easy and fast to use for most of the work to be done.

The result was code with a wide range of built in optional capabilities, but only those actually needed made fully usable, and others that proved unnecessary or ill-advised but harmless left in place. Parts were organised to allow easy upgrading if that proved necessary. In key places care was taken that the basic algorithms were efficient, and changes to the code to achieve near-optimal efficiency would be straightforward.

Since the writer was the only user of the system, a number of awkwardnesses that were of no great practical consequence were left uncorrected. Similarly

provisional fixes were made to a few faults that left minor restrictions in operation, of no practical effect, but irritating.

Considerable work would have to be done to make the system satisfactory for general use, but no fundamental program changes would be required, only a rather time consuming tidying up. The major task would be writing full documentation. The system would then be entirely adequate to process real rather than synthetic data. It is not intended to do this as there is no demand for its modest capabilities: most image processing requirements are now being met by dedicated systems.

2.2 Plotting System

2.2.1 Basic Method

The microfilm plotter is not a device intrinsically well suited to producing continuous tone images. It is possible to vary the intensity of the line drawn, by varying the beam current in the CRT, which varies the width of the clear line produced on the film. Unfortunately, because the film used gives high contrast, and its normal use does not demand close control of photographic characteristics, it is not adequate to use this intensity control for plotting images. The transfer characteristic would be highly non-linear, and even if this were to be corrected in the plotting program, the limited range of 30 intensity levels would then become totally inadequate as some of these would be very widely spaced in terms of the input data. Also the instability in the characteristics would make the maintenance of satisfactory overall linearity rather uncertain.

It not being feasible to change the film, the alternative was to produce 'half-tone' images, with each pixel represented by a symbol of size proportional to the value to be represented. This would have an intrinsic linear characteristic and would be stable. Its success would depend on the very high spatial resolution available on the plotter. The plotting beam could be positioned to any of 16,000 positions across the frame, and 12,000 on the other axis. Unfortunately the standard Imperial College graphics software sets an option on the plotter that causes lines to be drawn in steps of two basic plotter increments (which speeds up the plotting). This was judged to still be adequate, and a 'half-tone' method was chosen. Tests showed that the incremental accuracy of beam positioning was very good indeed, and there were no errors that would be significant in this work.

There was one slight defect revealed, from a different cause. When a uniform bright field was plotted, some faint dark patches were seen, presumably corresponding to less sensitive patches of phosphor in the CRT, or marks on the faceplate. Normally images were plotted as bright features on dark background, in which case the marks were negligible. During this work the CRT was changed, and later marks were not seen even on bright backgrounds, though no specific tests were made.

The simplest possible way to plot a pixel is as a single short line of appropriate length, and this was chosen as the basic method. It has the advantage over more

complex forms of keeping the length of the plot file to a minimum, which is significant because there is a system limit on the length file that will be sent by line to the site with the plotter. In comparison, folding a longer line so it builds up a solid rectangle gives better intensity resolution, and less loss of high spatial frequencies, but results in a much longer plot file and less inherent linearity. By turning the line elements slightly out of the line of pixels it is possible for each element to overlap the adjacent ones without interacting. Each pixel line element can thus be made longer than the spacing between pixels, which improves the resolution in intensity. A limiting length of twice the spacing prevents local high values from contaminating distant areas in the image, and appears as a sharp saturation in the intensity transfer characteristic. This method is shown in Fig.2.2a.

A disadvantage inherent in half-tone images is that as the plotted element gets larger, the change produced by a slight change in data value is no longer concentrated at the nominal position of the pixel, and this causes a loss of higher spatial frequencies from the image. With a line element the change is by an increment at each end, and when the line length is equal to the pixel spacing, the frequency response falls to zero at the ~~pixel~~ Nyquist frequency (in the direction of the line). At the Nyquist frequency the response is 71%, which is not usually a significant loss. Note that if the line element is not centred but has one end fixed there is no drop in response, but there is a phase shift that, for this work at least, would be worse. Allowing the line elements to become longer introduces a loss that may become serious unless care is taken.

In this work images were kept generally well below the peak brightness, and the upper part of the range was used to handle peaks of intensity without saturation. When a bright background was used, it was set well below the maximum level.

Because the start and end positions of each line element are converted to positions on a discrete grid, the intensity to be produced becomes quantised. And if, as in this work, the positions are subjected to an unknown scaling before they are quantised, the end positions are quantised independently. In fact, the start position is quantised, and then the change in coordinates is quantised into double size steps. The result is that the intensity produced on the plot for a fixed input depends slightly on the position of the pixel, but that at any position there is no average error, averaging over input values.

This would be a totally unacceptable defect, but it is easily corrected by adding a pseudorandom jitter to the input, uniformly distributed over a range corresponding to the step size into which the line length is quantised. Further minor improvements are effected by adding jitter to the element position and another component of jitter to the input value. The dramatic improvement that is effected is shown in Pl.2.2a.

2.2.2 Basic Program

As an aid to understanding the working of the plotting program some general points will be made, and then it operation described when plotting a continuous tone image in the normal, simplest fashion and ignoring all other options.

The system uses labelled common blocks to control its operation (blank common is used for buffers). All the parameters are set up in the blocks and then the plot is produced by a call to subroutine EXPLOT which has no parameters.
/KON/ holds the parameters controlling the scaling of data and the details of plotting a pixel in each of the major modes. Only the first part is used in basic operation.
/PROCOM/ holds parameters selecting options and most of those describing image dimensions and subdivision, and variables indicating current positions.
/LENGTH/ holds those parameters describing line lengths.
/STATUS/ holds details of the generated plot, and parameters for such things as error handling and time limits.

Many of the parameters used to select options each control a number of related but independent choices. Usually each choice is made by digits in specified positions of the parameter (represented as a decimal integer). In basic operation all of these parameters are preset.

Within the plotting routines positions and parameters are often initially given in integers which approximate to the integer coordinates needed by the plotter, and then converted to Fortran real numbers as needed by the standard graphics routines. This was intended to facilitate conversion to a more efficient form, bypassing the graphics routines.

Data values are also represented as integers, another choice made for reasons that in retrospect don't look very convincing. Since data is read in by routine NEWLN as reals, typically about unity in magnitude, it is multiplied by 1000 before converting to integer. To handle smaller input values, an option providing other scaling factors was provided.

Plotting is caused by a call to subroutine EXPLOT, and proceeds controlled by the parameters preset in common. This calls PRORDY to perform certain initialisations, in this case to initialise the data input routines by a special call to LNDATA and to set up a frame of microfilm with a call to MFPAGE, which will generate a border and a grey scale if required (using MFGS).

EXPLOT then enters a loop producing lines of output. A call to PROSEQ establishes the type of the next line, which may be a line forming a top border, a line of image, a line of bottom border, or a final dummy. The line is then plotted by PROLN. It generates a border line with LNBORD. For an image line it gets a line of input data with LNDATA (which uses NEWLN to read a record from the input file) and forms it into a line of output with LNOUT, which adds the side borders. (LNSPEC and LNSCAL have no effect). A generated line, border or image, is plotted by LNPLOT, which calls MFZ to plot the line as continuous tone on microfilm.

If the line type is a final dummy, a call to BLKEND adds a label with identity numbers to the frame. The loop then terminates, EXPLOT reports to the terminal the processor time used and the current length of the plot file, and control returns to the command interpreter.

Subroutine MFZ moves the previous start position down the frame by one grid interval and performs various initialisations and parameter conversions. It then loops over the data values in its input array, ignoring the first which is used (elsewhere) as a layout control character. For each the position is stepped along the line. If the value exceeds the range permitted for data, a symbol is plotted by a call to MFSYM unless the value represents a blank. Otherwise the value is data and is

scaled to give the length for the line to represent the data point. Pseudorandom jitter is added to this and to the nominal position, and the line is plotted at an oblique angle. If the data point is the first, a dummy point is plotted first at its left with the beam off, to avoid any end effect.

2.2.3 Refinements

The routines include many extra features: only the major ones will be discussed, and them briefly.

Some have been mentioned. A grey scale can be plotted down the right of the frame, either a continuous wedge or stepped, and optionally labelled with corresponding data values. A pair of identity numbers are written on the frame: these are manually entered to identify the frame, and have proved indispensible. A border is produced round the image area: the width can be varied, and it is basically blank with symbols round the outside. They can be in various patterns; the one normally used serves to define the exact extent of the image area and can act as registration marks. A sample frame is shown in Fig.2.2b.

Overall format can be varied. Either the full plotting frame can be used, or plotting restricted to the area of a standard 35mm frame. The partial frame border can be omitted. The plot can be compressed vertically by a factor of two, intended for optical expansion with an anamorphic lens system and allowing better intensity resolution for a given total image area in pixels.

The input data can be scaled logarithmically, inverted so it plots dark on light, or converted with a table lookup (which might be set to represent the transfer characteristic of a photographic plate).

The actual plotting process incorporates two provisions for increasing the dynamic range (that is, improving the resolution in intensity). It can break the input range into three, and cover each part with the basic method but at a different beam intensity: with correct constants the three ranges can be fitted together. And it can allow the line representing a pixel to increase beyond the maximum length by folding it back to form a horizontal V, as shown in Fig.2.2c. These can be combined. Other variations, such as lines running nearer vertical than horizontal, and elements staggered not skewed, and possible but were intended only for tests.

In the more general part of the program, the main complications are to handle dividing the plot into sections, combining plots, and adjusting the sampling intervals: these will be discussed later. One other feature is the ability to use, instead of an external file of data, a simple internally generated pattern, which allows the plotting system to be tested on its own.

2.2.4 Contouring

The subroutine MFC, called by LNPLOT instead of MFZ, enables contours to be plotted. While neither the efficiency nor the quality of the result is the best possible, both are adequate and it requires a minimum of new code.

The data is contoured a line at a time. Each line is processed along with the preceding one, saved by MFC on the previous call. The pair are taken 4 adjacent points at a time, and each square cell so defined is contoured by MFCL.

The range of the 4 data values forming the cell is tested to see if it crosses any contour levels. If it does, the crossing points are linearly interpolated along each side, and straight lines are plotted between crossings of the same level.

The contour levels are set by scaling and conversion of input data as for MFZ, so they are regularly spaced, linear or logarithmic, unless a conversion table is used. The number of contour levels can be varied, and they can be displaced by fractions of an interval. The contours are not labelled, but a proportion of the levels can be plotted with a heavier line.

The number of levels plotted in a cell can be limited to avoid unduely close spaced contours blocking up. If the number crossed is excessive only a proportion will be plotted. This gives a messy, irregularly interrupted appearance to contours in crowded regions, but suffices to keep them readable.

A special border symbol is recognised by MFC and is not plotted but defines a box round the plot, which is drawn by MBX.

In normal use, contours are logarithmically spaced at intervals of 0.5 magnitude (a ratio of 1.585). If a 'grey scale' is selected it shows the contour levels used, and the labels give the log of the level relative to the base value. An example is given in Fig.2.2d.

2.2.5 Line Printer Output

Image output on a printer is necessarily of low quality, but the low cost and rapid availability of hardcopy (typically 1 or 2 hours instead of 1 or 2 days on the system used) is potentially useful. In practice, contour output on a graphics terminal sufficed for checking generated images, and the printer output was not used.

Two types of output are possible; either an approximation to continuous tone using overstruck characters, or the equivalent of contouring with the areas between levels distinctively shaded. Both are produced in essentially the same way, with a routine LPRTZ called from LNOUT. The routine converts the data values in the line to be printed into symbols by scaling onto a range of 128 and looking up a table in /KON/. It then calls LPRT to print the line of symbols. If more than one character is to be superimposed LPRT overprints lines using characters taken from successive positions in the words of the input array (each word can hold up to 10 characters).

To be useful it is necessary to squeeze the image vertically to compensate for the printer pitch being greater vertically than horizontally (6 or 8 per inch against 10). Also images need to be adjusted in size either to make full use of the page or to fit them on it. These functions are provided by calls to LNSPEC, LNDATA, and LNSCAL from PROLN. These re-sample the data at the required intervals, rather crudely using linear interpolation or grouping and with some restrictions on ratios, but adequately for the intended low quality uses.

Only the table for the equivalent of contours was set up. The quality was very satisfactory: an example is shown in Fig.2.2e.

Another form of printer output is included. A call from LNOUT to LDMP is used to produce a numeric listing of the image, for test purposes only.

2.2.6 Sectioning and Merging

A great deal of the apparent complexity of PROLN, LNOUT, and LNBORD especially is a result of providing the ability to efficiently plot an image as a series of overlapping smaller images. The intention was that large images could be effectively handled on plotting devices of less than ideal resolution. In practice this was tested but not used, and in retrospect the ability plot a part of the image area would have been ample. The loss in efficiency incurred in reading the input file repeatedly as each part of the image was plotted in turn would have been acceptable, and the image sectioning capability could have been coded as a control routine to set up and plot the successive subimages.

The basis of the method used is to divide the image into vertical sections, and into horizontal blocks. Each input line of data is converted into a line to be plotted, complete with borders, written to a file for each section. At the end of a block the last lines of data that form the overlap are written to a scratch file and read again at the start of the next block. The actual plotting of the section files can be done after each block is processed or at the very end. The section files can be replotted with a different method or scaling with a minimal overhead.

A related source of complexity is the ability to combine successive plots on a single frame of microfilm, which is simple, or on a line printer, which is less so. This provision for merging printer plots uses calls to LMERGE from PROLN and LPRT to build up an output file by copying between two scratch files, and a final call to PRORDY to actually output the scratch file. It was intended to allow pages or frames of small images to be built up as indicies to the full size plots: again, this feature was unused.

2.2.7 Parameter Setting Routines

To enable the general plotting routine EXPLOT to be used, routines are provided to initialise the common blocks and enter parameters into them. Their names begin SET.

Some of these set common variables directly. SETLIM sets limits in /STATUS/, and initialises it on the first call. SETSYM enters a symbol table into /KON/. SETKSD and SETKSX are alternates for setting /KON/, SETPSD for /PROCOM/ and /LENGTH/. SETVAL will set a single variable in any of the four.

Others use these to set up different functions. SETPRZ zeroes /PROCOM/, and SETCOM (using SETCON and SETPRO) performs a complete set-up assuming default parameters for those not supplied, ready for subsequent modification. SETLAB, SETBOR, SETMOP, SETSEC, SETDIM, SETLOG and SETBOX enter their parameters to control individual functions.

SETZ and SETC perform complete initialisations for standard microfilm continuous tone and contour plots. PLOTZ and PLOTC are equivalent routines that include calls to EXPLOT to do the plotting: PLOTC also includes a switch to log contours and the proper border.

These were adequate for testing, and the PLOTZ and PLOTC routines sufficed for most of this work, but they are not comprehensive. Others are needed to make functions such as printer output and sectioning conveniently usable and to exploit the flexibility of the plotting routines. They are easily written as required and incorporated using the generator program.

2.2.8 Making continuous tone prints

The 'half tone' microfilm plots are entirely satisfactory for direct examination, but for reproduction it is necessary to suppress their structure, converting the plot into a true continuous tone image. This requires an optical filtering process, and inevitably leads to a slight loss of high spatial frequencies. The loss is minimal if the plot is copied using a lens fitted with a square aperture stop, just sufficiently out of focus for the square blurs to merge.

This method was tried, but the lens used was not of sufficiently good quality and the system was of inadequate mechanical precision, so that the potential advantages of the method were lost. Instead, the plots were copied onto film, slightly out of focus but with a normal circular aperture. The focus was adjusted visually to just suppress the dot structure as seen on the screen of a 35mm SLR. The negatives still showed a dot structure, though much reduced. Defocusing in enlarging provided further reduction. It was kept to a minimum, and a slight line structure was accepted in the prints, to avoid undue loss of high frequencies.

The prints made by this method retained all required high frequency detail. They were generally similar in appearance to the original plots, but no special care was taken and tone reproduction will be substantially non-linear. As reproduced here, they have been through a second copying process, further increasing the distortion. They are essentially illustrative, and may not be suitable for quantitative examination.

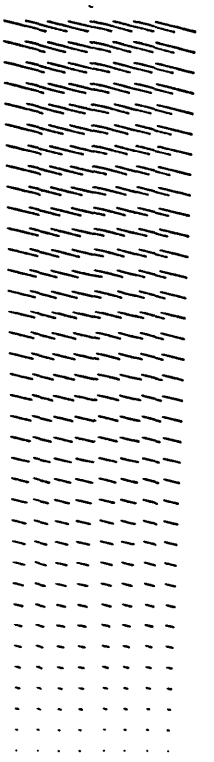


Fig.2.2a Method of plotting continuous tones.

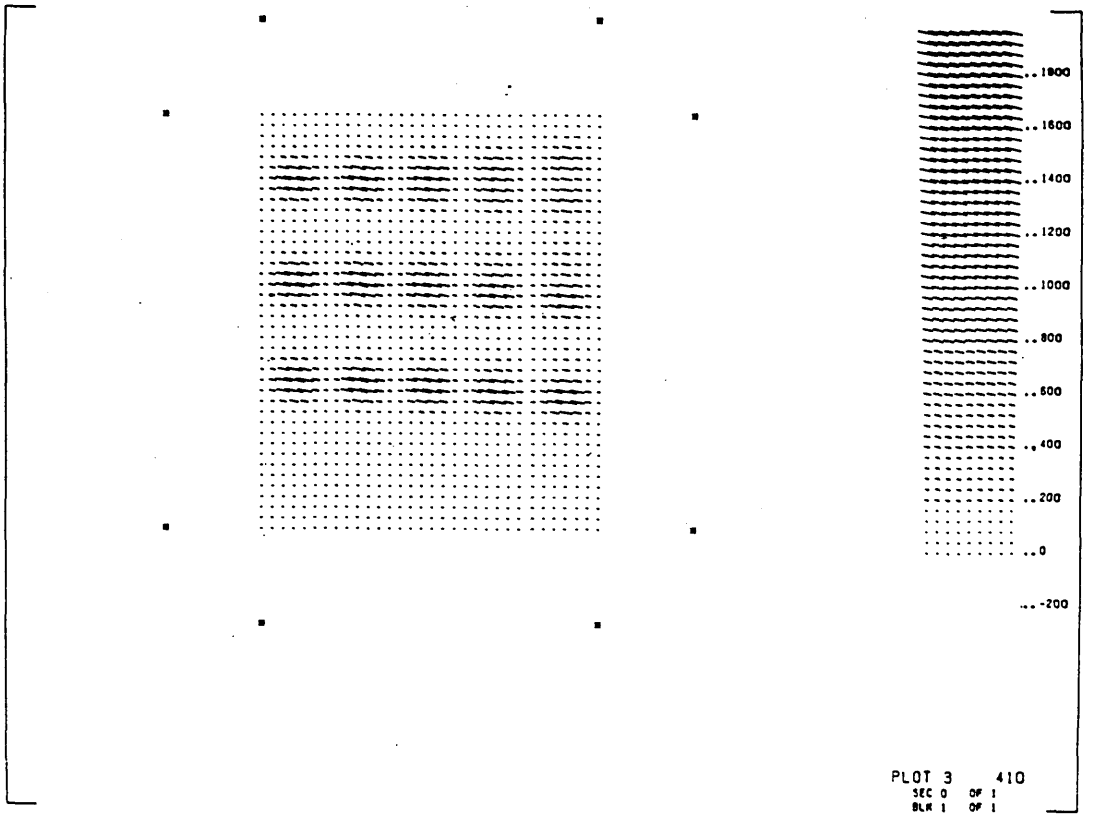


Fig.2.2b Sample frame with continuous tone plot.

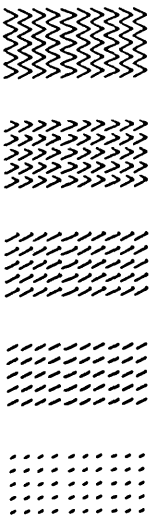


Fig.2.2c An alternative method of plotting continuous tones.

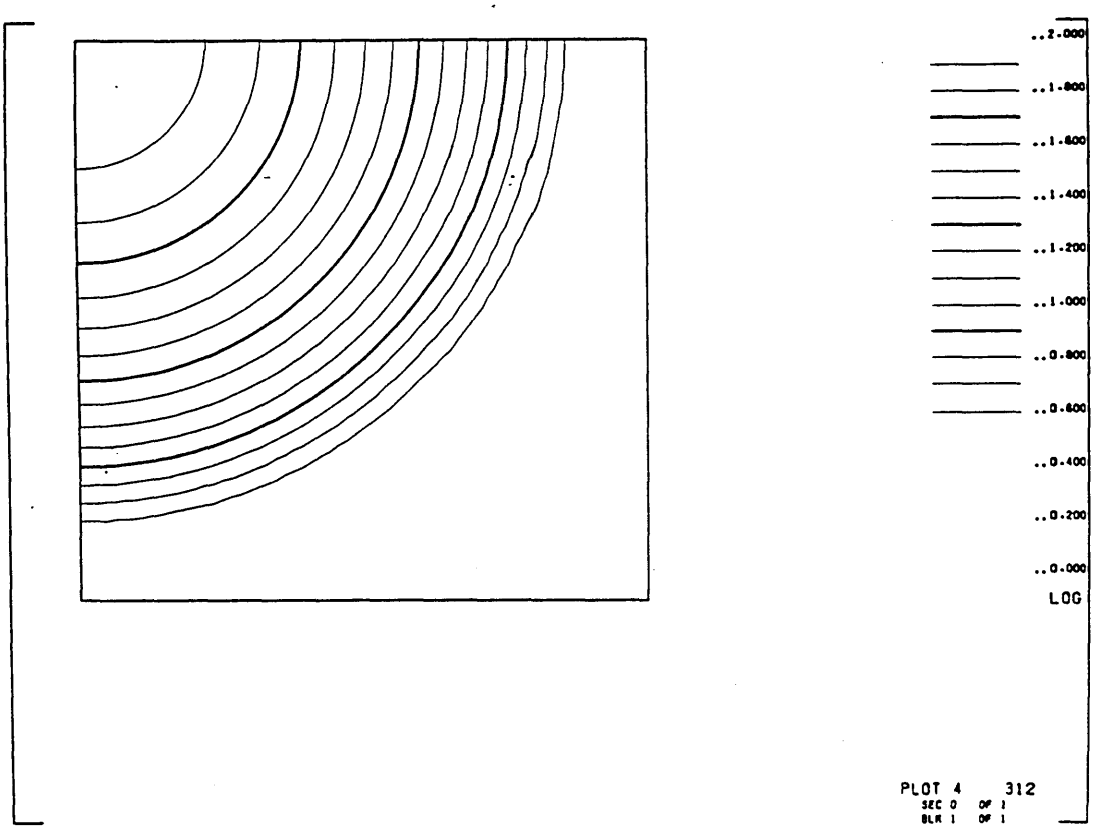


Fig.2.2d Sample frame with contour plot.
The image is one of the built in test patterns.

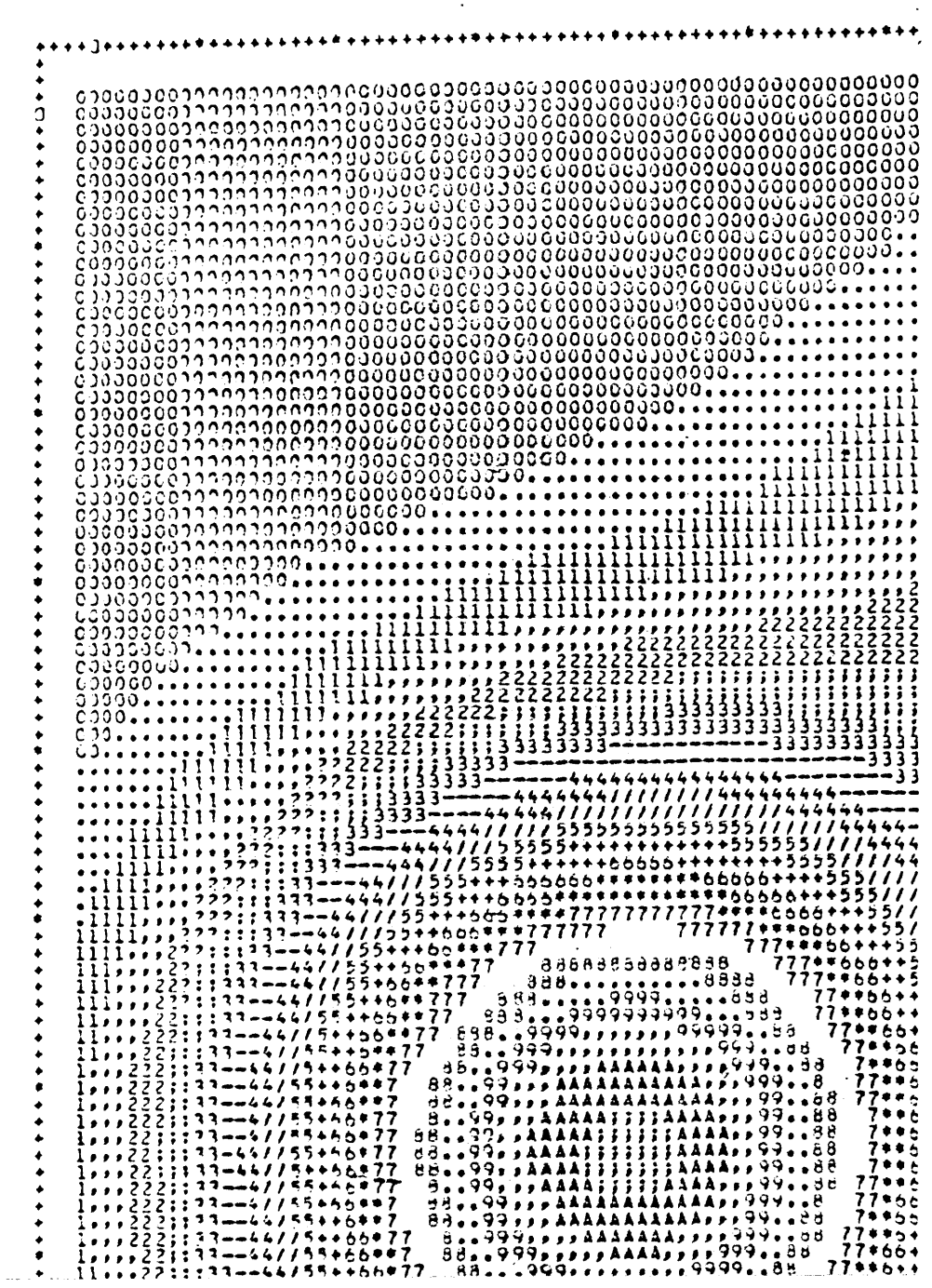


Fig.2.2e Sample line printer output.
One quarter of a plot is shown. The image is another built in test pattern.

2.3 Image Generation System

2.3.1 File System

The routines of the image generation system operate on binary files of real numbers, each record representing a line of an image. The files are referenced by their Fortran 'unit numbers', and the dimensions of the image in each are recorded in the common block /FT/ and made available by FLDATA. All file I/O is handled by routines GETLN and PUTLN to allow easy upgrading of the file system, and the necessary extra parameters are provided.

When a pre-existing image file is to be processed, a routine FLSIZE is used to set its dimensions in /FT/. When a file is generated internally, a routine FLSET is used to set its dimensions relative to another file, which is by default rewound, and remove any existing contents.

File number 0 is used as a dummy with nominal dimensions 0,0 and causing no file operation to be attempted. File number 1 is a similar dummy for which dimensions can be set in /FT/ and GETLN will return a line of zeroes. Their use increases the flexibility of file processing routines. For example, FLSET used with file number 0 sets a file to an absolute rather than relative size; and a routine to add noise while copying from one file to another, used with 1, generates a file of just noise.

2.3.2 Star Field Generation

The basic image generation routine is STRGEN, which produces the image of a field of stars added to an existing image to give a new one. The stars are specified by x,y coordinates and peak brightness in a table; the form and width of their profiles by parameters of STRGEN.

The image is built up a line at a time. A cutoff radius is initially set applying to all the star images, and for each line, each star is tested to see if it is within that radius of the line. If it is, a cutoff radius is calculated for this star. If within this radius also, the corresponding range of points along the line is selected, and for each the distance from the star position is tested. Only if this is less than the cutoff radius is the star's brightness at that point calculated and added into the line.

This is a simple but efficient organisation, suitable for large images and large numbers of stars. A further slight improvement is made by detecting when image generation has passed a star's position, and altering the table to move the star a long way beyond the bottom of the image: the star will then be skipped at its first test on each subsequent line. The table is restored at the end of the routine.

Note that the lines are assumed to run horizontally left to right and in sequence from top to bottom (as in all the image generation and plotting routines), but that the y axis runs left to right, the x from top to bottom, with the origin at the top left corner. This convention is used only here and in the peak location routines. The length of line is taken as 1.

There are several main options. The star profile is normally gaussian, but it can be flat topped, cosine squared of r or r squared, or generated by one of two separate subroutines. The star images can be forced to be centred on a pixel. The image can be as small as possible, either just on the nearest pixel, or on the 4 nearest pixels with total brightness as the value given.

2.3.3 Table Set-Up

The table specifying the star field is a one dimensional array, usually with 4 elements per star though only 3 are used, and is passed as a parameter. Routines are provided to produce the table. Most simply, STRCLR, STRSET and STRMOD clear the table, add a new entry and modify an existing entry, while STRLST lists the table.

Random star fields can be built up by initialising with STRFLD, followed by STRINS to add stars with a given minimum spacing. STRDBL adds double stars of specified parameters, except position angle which is random. These use subroutine STRTST and common /STRCOM/.

The most general routines are GRDCLR and GRDDEF, which clear the table and define a rectangular grid of positions (which may include a random component), and GRDSET which will set or modify any column of the table in many ways, including setting up various types of series, adding pseudorandom numbers of specified range, and taking the antilog. GRDRND will shuffle the entries in a one column. Together these will set up many types of starfield. Fields of double stars can be generated by setting up the table with 8 elements per entry, 6 being used for the elements of each double: GRDDBS will then convert it to the corresponding table of single stars, two entries of 4 elements for each double star.

2.3.4 Elliptical Galaxies

The starfield routine STRGEN can also generate fields of elliptical galaxy images. A routine ICOMPF handles both the setting of cutoff radius and the calculation of the brightness at a point in a galaxy image, and is passed the table of galaxies. The table has 12 elements per entry, and is set-up in the same way as tables of stars.

An initial call is made to ICOMPF to find the maximum cutoff radius of any galaxy. This is handled by a call to INITCF which initialises the table by converting the galaxy parameters to the form needed for efficient computation, including setting a cutoff radius for each and returning the maximum. Calls to

ICOMPF are then either for the cutoff radius for a particular entry, available directly from the table, or for the brightness of a point, which is readily calculated.

The galaxy parameters are two coordinates, integrated brightness, position angle, effective radius, and ellipticity. The effective radius is the geometric mean of the values on the major and minor axes, so the effective area does not vary if the ellipticity is altered.

The only problem is that the profile used has a sharp central peak, which with a small effective radius is narrower than the sampling interval. This is handled by using for the brightness, when the point is very close to the centre, an approximation to the average brightness over a central square pixel. This is at least adequate, provided in critical cases the option is used that makes STRGEN centre the images on a pixel.

Stars can be included along with galaxies, indicated by a very small effective radius. They contribute to the 4 neighbouring pixels, the total brightness being correct.

2.3.5 Noise

Routine NOISE copies a file, adding noise with a gaussian distribution. Normally the noise has a constant standard deviation, but it can be optionally made proportional to the square root of the input value at that point. This allows it to approximate to the poisson noise of photon statistics: not exactly, but well enough when the noise level is low. The noise is pseudorandom, and a parameter specifying the 'seed' enables different fields of noise to be repeatedly generated.

The random deviates are supplied by RDG. The only unusual feature is that it takes in effect two seed values, one being the line number, so a part of a noise field could be generated without having to calculate all the previous lines: this was not used. It can also return uniform deviates or a random sign: these were for use in other programs. The algorithms are standard.

2.3.6 Filters

The routine FLTXY provides simple spatial filtering, applied as successive one dimensional filtering (by convolution) horizontally and vertically (along and across lines). Each filter is normalised and can be independently specified. Each can give either uniform weighting along its length, or gaussian weighting about its centre. Their normal use, in combination, gives either a uniformly weighted square filter or a circularly symmetric filter with a gaussian profile. The separation into two one dimensional filters restricts the types of filter possible but gains typically an order of magnitude in speed.

A one dimensional array, passed to the routine as a parameter, is used to hold the filters, the input and output buffers, and a cyclic buffer of lines being processed. The size needed depends on the image and filter sizes: it is large, but usually much less than the entire image.

The filters are set up and normalised, the cyclic buffer zeroed, and other initialisations made. Then each line of input is read in and filtered along the line with the result being put into the cyclic line buffer. For each line, the vertical filter is scrolled down to keep it aligned with the contents of the cyclic buffer, and if a

line of output is to be generated a weighted sum of the lines in the cyclic buffer is formed in the output buffer and then written to the output file.

Normally the output image is smaller than the input by about the size of the filter. Alternatively, it can be made either the same size or larger. The outer part of the filtered image then has less terms contributing to each point: for a uniform input, the output will fall off at the edges. As an option, this can be avoided, with the filter being effectively renormalised at each output point to compensate for the reduced area of filter actually being used.

If the required filter cannot be decomposed into orthogonal components, a two dimensional filter has to be used, and this is provided by the modified routine FLT. The filter is read into the buffer array from a file. The cyclic line buffer is retained, and the filter is again scrolled to keep in step.

2.3.7 Peak Finding

A simple system for finding peaks in an image is provided. This was intended for very limited uses and is not entirely reliable on images without clear isolated peaks. The routine PEAKS invokes the full process, using a number of routines beginning PK. They will not be discussed in detail.

PKFIND is the routine that actually finds the peaks. PKCEN then attempts to interpolate exact positions and estimate the true peak value: the method is only correct for circularly symmetric peaks.

The system works adequately on star images well above noise, and that have been through a filter at least approximately matched. For use on more general images the results should be treated with caution; and the algorithms used need improvement.

2.3.8 Utilities

There are three other routines operating on image files.

FLSUM adds two images to give a third, multiplied by weighting factors and adding the value of another parameter to all points. A border can be trimmed off the final image, of equal width all round.

FLSTAT reports the statistics of a file, the maximum and minimum values, the mean and the standard deviation. Optionally it also produces a distribution over 20 intervals either linearly or logarithmically spaced.

FLLST lists the data forming a selected part of the image, and is used only for testing.

2.4 One Dimensional System

2.4.1 Basic One Dimensional System

This generates, processes and displays relatively short sequences of data. The length of such sequences (or linear arrays) is a constant preset at program generation: the lengths used have been 51, 101, and 201. The linear arrays are referenced by number and are accommodated in a large one dimensional array (the length used has been 10404) passed as a parameter to subroutines. Often they are regarded as representing one half of a symmetrical profile.

Some subroutines operate on groups of three adjoining linear arrays ('triples') as a single sequence of data, referenced by the number of the middle array. This allows a filter, operating on such a longer sequence, to produce an output sequence of full normal length.

The linear arrays can also be grouped as the lines of square, two dimensional arrays. Only a few of these square arrays, also referenced by number, can be accommodated. They are used to represent one quadrant of a circularly symmetric image defined by a linear array as the radial profile.

A function JV is used to check that reference numbers, up to 4 in one call, specified as referring to linear or square arrays, are valid. It can sometimes be used to check other parameters as well.

A function RXT is used to obtain the value at some position in a linear array. It will accept any position, using linear interpolation if necessary; assumes the array is half a symmetric set of data; and returns zero if the position is beyond the end of the array.

Routines SAVE and REST will copy the contents of the data area to, or restore it from, a file. This allows a job to be interrupted and continued at a later terminal session. They can be used selectively.

Numerous routines provide straightforward data manipulations such as adding pseudorandom noise, generating gaussian profiles, adding arrays, and convolving them, and will not be listed in detail. One routine, FNC, provides a considerable range of mathematical manipulations, selected by a name parameter. A function PSUM numerically integrates the point by point product of two linear arrays,

weighted with some power of the position along the array, and is used especially in evaluating the performance of circularly symmetric filters.

Only a few operations are provided on square arrays. One, RD, generates a square array from the radial profile given in a linear array, the radial scale being adjustable. Another, CONSQ, convolves two square arrays to give a linear.

2.4.2 Fourier Transforms

Routine FTRANS provides a discrete cosine transform of a linear array, which is the entire fourier transform if the input is assumed to represent half a symmetric set of data. It uses the most direct and inefficient method, which was simplest to implement and was considered acceptable for the short lengths of data to be used and the low frequency of execution. The transform is defined in such a way that it is its own inverse.

The routine is provided with an extra parameter that allows the width of the transform to be altered. This is done, in effect, by appending a length of zero data to the input, and discarding the excess length of output. Expanding the transform in this way allows its shape to be seen more clearly when displayed graphically. Because of the truncation of the output, the transform is no longer exactly its own inverse, but used carefully so the truncated part is near zero the error is imperceptible.

A second routine FTWO uses this to provide the fourier transform of a circularly symmetric image, input and output being the radial profiles. It uses a square array to hold a quadrant of the image formed from the input radial profile, which is then projected into one dimension with LN and this is transformed with FTRANS. The overall effect is equivalent to a Hankel transform, that is, the analogue of a cosine transform using the zero order Bessel function. It is unfortunately susceptible to numeric errors introduced in the re-sampling needed to convert the input to a square array. These would be avoided by the direct evaluation of the Hankel transform, but the method used was simpler and considered adequate.

2.4.3 Image Generation

Two routines are provided that generate standard image files from one dimensional data.

Routine SPEC forms the product of two linear profiles, one along and one across the lines of the image. The two profiles are scaled to fit the specified image size. This was devised to construct images resembling photographic spectra: the spectral data being given a transverse profile that is typically gaussian or rectangular.

Routine FLTGEN produces a square image that is a central portion of the entire image represented by the square array given as input. This enables a filter to be constructed from a radial profile for subsequent use by routine FLT in the image generation program.

2.4.4 Interaction and Graphics

A necessary complement to the manipulative routines are those that display linear profiles as graphs, allow measurements to be made on these graphs, and modify the data displayed according to the graphical input using the crosswire cursor on the terminal. These are rather long, of no great relevance to the processing capabilities of the system, and of little intrinsic programming interest in this context. Accordingly they will be neither discussed nor listed, but it should be noted that a routine to display a linear array graphically is an essential part of the system.

An extensive set of general purpose graphics routines was also written, which gave great flexibility in turning linear data into finished, fully annotated graphs that could be developed at the terminal and then plotted on microfilm. While this was a very useful addition to the system it has only marginal relevance to this work, and this too is omitted from the program listings.

2.5 Command Interpreter and Generator Program

2.5.1 Command Interpreter

The command interpreter uses a loop to read and execute lines of input. This loop is entered half way down the program (at a statement labelled 9950) because it has been rearranged so all the conditional statements and subroutine calls that are peculiar to a particular application come at the top. This allows a file containing them and the program statement, declarations and initialisation code that also depend on the application to have the file with the constant part appended to give the complete interpreter.

After a line has been input it is decoded and then processed to implement special actions such as requesting confirmation and copying to a file. Execution then jumps to the long sequence of conditionals which searches to match the command entered as the first item on the input line. When identified appropriate action is taken, normally a subroutine call using parameters supplied on the input line, but arbitrary code can be executed. A command beginning with a $ character is one controlling interpreter functions and is processed in a similar sequence of conditionals at the top of the constant part of the interpreter. If a command is not recognised, failure is reported and if the program is not being used interactively it is stopped.

2.5.2 Input Line Decoding

The input is read as a string of characters which is then decoded by routine CONLIN, which uses LTOK, LGAP, and CONTOK. The string is broken up into groups, blanks or commas acting as separators. Each group (or token) is converted into a numeric value if possible, decimal points and E format being recognised, and a text value is set to blank. If not, the numeric value is set to zero, and the text value is set to the input characters packed into one word (up to 10, blank filled). CONLIN returns the numeric and symbolic values from all the input groups in two arrays, filled out with zeroes and blanks, as well as the total number of input groups and the number that were not numeric.

In the command interpreter the array of numeric values is converted from real to integer, in a second array. Thus both types of numeric value are available for use as parameters in subroutine calls, as well as textual values, without imposing any restrictions on the input format.

Unlike standard numeric input, the number of values that have to be entered is not fixed. It thus allows for routines taking different numbers of parameters. If fewer values are given than needed, the rest will be zero, which means that if parameters are placed at the end of the list they may be regarded as optional and omitted provided the routine is written to give a sensible default for a zero input.

2.5.3 Command Interpreter Functions

The power of the command interpreter is greatly increased by the provision of various extra functions. A number of minor ones will not be discussed. Note that the $TERM command must be given if the program is used interactively, or it will abort on errors.

Most important is the $FROM command, which causes input to be taken from a file, which can be specified by number. Input reverts to the original source when $END is read or the end of file is reached. This enables a file of commands to be prepared and executed repeatedly. It might be used to generate standard images as a basis for subsequent processing, or to construct a complicated graphical plot that could then be corrected or modified and replotted.

The $LOG command is to some extent complementary. It causes input lines to be copied to a file as well as processed immediately, normally omitting any interpreter commands. A file built up in this way can be executed with $FROM to repeat the same sequence of commands. $LOG can also be used to keep a file of commands entered that can be printed as a permanent record of work done.

Layout commands such as $PAGE and $TEXT write to a text file, often one being generated by $LOG or one receiving program output, adding new pages, dividing up the output, and adding annotations.

A number of file manipulation commands are provided, such as $REWIND, that enable files to be positioned, copied and listed. These are useful capabilities, especially during program development when files are not being correctly handled.

There is also provision for a help facility, available though $HELP followed by the command name. This causes the command name to be searched for as usual, but instead of calling a subroutine, it outputs an appropriate message. It is a desirable facility but this version is not well implemented and it was not used.

2.5.4 Generator Program

Producing the application dependent part of the command interpreter is the primary task of the generator program. As a secondary task it produces a procedure file that when executed combines the variable and fixed parts of the interpreter, adds the files containing the application subroutines, and compiles the result. Provision is also made for producing a procedure file to run the compiled program, including the use of libraries, but this is not fully implemented and the subroutines needed are dummies.

The input to the generator is a text file, processed a line at a time. This begins with a specification of the program statement and the files required. Then come any declarations and other statements to be put onto the start of the interpreter. The main part follows, listing the subroutines to be recognised by name and the parameters required by each, and including any associated help messages. There may also be names associated with sections of code to be included so as to be executed when the name is recognised. Finally there are lists by name of the files containing the required subroutines. Lines beginning with * will be ignored, allowing remarks to be interpolated in the input. Lines beginning with $ have a controlling function.

Certain of the control lines are recognised and processed independently. In particular, those beginning $S, $SL, $B, or $BL. These are followed by the lists of file names and are used to generate the commands in the procedure file which will add the corresponding files to the system. The normal one is $S, $SL is a version that assumes the files are already local, and the versions with B are for files already compiled, which are added after the compile command (and must follow all S versions). The file SLCON contains the routines used by the command interpreter and is always included: a check is made to prevent it being added twice.

Of the remaining lines, if the first begins $P it is used to generate a program statement. The unit numbers of the files to be declared are given after the $P and a program name. If a non-numeric value occurs, it is treated as a file name and is equated to the subsequent name, with I and O being expanded to INPUT and OUTPUT. If the first line does not begin $P, it is assumed to be a program statement to be copied directly, including any continuations, unless it starts the conditionals section in which case no program statement is used.

After any program statement some standard declarative statements are added. Input is then treated as lines of source code to be copied directly to the output, until a line beginning $ marks the start of the conditionals section. The lines are put into a format suiting the MNF5 compiler with the line number (K) option: the space, comment or continuation character that is to follow the line number must be given.

2.5.5 The Conditionals Section

At the start of the conditionals section, normally by a line containing just an initial $, a few standard program lines are output, beginning with a jump into the start of the loop, a labelled statement that is the destination of a jump at the end of the standard part of the interpreter, and the start of the long block IF statement that uses multiple ELSEIF clauses to process the different commands.

A line not beginning with $ is then broken up into groups of characters separated by spaces or commas. The first group is the subroutine name, and it is used in an ELSEIF line as the command to be recognised, and then in a call, conditional on a control variable, to the subroutine. Also conditional on control variables is a statement that produces a message echoing the parameters specified.

The groups after the name specify the parameters required in the call. If a group is R, I or K it specifies an element of the real, integer, or text arrays set up by the input conversion in the interpreter. Otherwise, the group is just copied as the parameter itself.

A line beginning $H or $I adds a write statement, again conditional on control variables and also on the character immediately following the H or I, that reproduces the rest of the line as a help message.

A line beginning $ followed by anything else treats the subsequent group of characters as a command name and outputs an ELSEIF line to recognise them. Following lines are treated as lines of source code, and are copied to the output. Another $ followed by a name produces another ELSEIF line, but a $E returns to the normal conditional mode.

2.6 Computing: Assessment

2.6.1 Size, Speed and Cost

The program comprises a total of 6000 lines of source code, exclusive of comment lines. The plotting subroutines are 2000, the image generation 1300, the one dimensional 1700, and the command interpreter and its generator 1000. There are also 4000 lines of comments, making a total of 10,000 lines. The entire source code occupies 240 kbytes.

The processor time required for image plotting or generation is nearly proportional to the image area. Approximate values will be given for a 200 by 200 pixel image.

a) Plotting a continuous tone image: 30sec
b) Plotting a contour map of an image, log contours: 30sec
   This is for a contour map about as crowded as is reasonable, with sparse contours it is faster: 24sec
c) Filtering an image, gaussian filter, sigma of 1 pixel: 10sec
   With a sigma of 4 pixels: 25sec
d) Filtering an image, externally specified filter as required for deconvolution.
   With filter 9 by 9 pixels: 45sec
   With filter 17 by 17 pixels: 150sec
e) Adding noise to an image: 12sec
f) Generating a star field image, with 40 stars, sigma of 2 pixels: 2sec
   With 250 stars: 9sec
g) Generating a field of 10 elliptical galaxy images, with no cutoff: 180sec
   With cutoff such that on average there is no overlapping: 20sec

Note that elapsed time depends on the load on the computer; typically, working in the evenings, it has been two or three times the processor time.

The internal costing by the computer centre to the department is based on a resource unit which in this work bought an average of 50 seconds processor time, including associated memory and I/O usage, but excluding the terminal connect time which was charged at about ½ unit per hour. A unit costs 35p in the daytime,

but in the evening it is charged at half price. Thus the evening cost in pence is about one third of the processor time in seconds, and to plot a 200 by 200 image costs about 10p.

There is in addition a charge for generating microfilm of about 3p per frame, plus a fixed charge of about 17p per file plotted. Since only one 200 by 200 image can be accomodated on a file the cost of plotting a continuous tone image 200 by 200 is about 20p, and the cost is again proportional to the image area. Contour maps cost less as more can be fitted on one plot file: 7p would be typical.

Thus the total cost of plotting a 200 by 200 image would be 30p for continuous tone, or 17p for contours. The cost of generating a starfield image of that size, adding noise, and filtering it, would typically be 10p. These are very reasonable. Only deconvolutional filtering is inclined to be expensive, at 50p with a 17 by 17 filter, which is a typical size.

This costing, while relevant in an academic environment, does not represent the true cost. Units are charged to commercial projects at £3, and their true overall cost is said to be £10, nearly 30 times the internal cost. This is misleading as it includes the cost of a wide range of services and a computer centre staff of 120.

To attempt to find a meaningful true cost of the computing power used, albeit very approximately, consider that a VAX11/750 installation with about 20% of the speed of the Cyber 174 used costs, very roughly, £100,000. Writing it off over 5 years, allowing 15% of its cost per annum for maintenance, and including the cost of someone to look after it, the cost would be roughly £50,000 a year. If 60% of this is charged to a daytime (8 hour day, 5 day week) time-sharing service (the rest to batch, evening and weekend work) and the processor is on average 35% used (to ensure a good response), the cost of the 700 hours processor time used in a year under time-sharing is £30,000, or £40 per hour. A unit on the 174 would be approximately equivalent to 4 minutes processor time on the VAX at a cost of just under £3. Even if the system was heavily loaded, it would be well over a pound.

It seems reasonable to say that the true computing cost to generate or plot an image, for work done in usual hours, should be 10 to 15 times higher than those quoted above. And it seems probable that much the same applies to the microfilm plotter: the academic charging is based on just the cost of the consumables. A representative true total cost for plotting a 200 by 200 image might well be something like £2.

The overall usage of computing resources on this work has been 8 hours processor time and 425 hours terminal time, charged at a total of about £170, and 1000 frames of microfilm charged at £95. Program development and testing took about 2 hours processor time and 320 hours terminal time out of the totals and accounted for about £80 of the cost.

2.6.2 Restrictions

The system is subject to one inherent limitation in its capabilities, and that is the noise level introduced into continuous tone images by the finite resolution of the microfilm plotter. This is proportional to the picture resolution used. At 200 by 200 pixels, and without using the techniques provided to extend the dynamic range (which were tested but not employed in this work), there is in effect a quantising error of $\pm 0.8\%$. Other sources increase the noise slightly, so it is about equivalent

to a quantising error of ± 1%, which correponds to quantisation into 50 levels. The added jitter masks the quantisation but with some increase in noise. This is generally adequate in view of a limit to image area of 48,000 pixels imposed by the restriction on length of plot file transmitted.

There are other limitations which are arbitrary. The plotting program will not handle lines of more than 256 points because of the length buffer provided. Another array limits the number of stars in a field, but at 1249 this is not usually significant. The same array is used as a buffer in the filter routines with more serious effects. With a 200 by 200 image it limits the filter size to about 21 by 21, with 100 by 100 a filter of at least 35 by 35 is allowed. This array is defined in the script for the generation program and is easily made larger. The number of files available to a program is similarly defined, normally about 6 or 8 being provided for images, and is equally easily increased.

The program for one dimensional data is substantially less general. Both the array used for data, and the length of each data item, are fixed in the generation script. The length often had to be changed, requiring the program to be rebuilt.

There are also some outright defects. One, the fact that pressing 'return' alone when the command interpreter is waiting for input stops the program, seems an unavoidable consequence of the way Fortran handles input. Others are minor but correctable, left because they were of no consequence in this work. The worst of these is in the scaling of data in the contouring routine: the contouring goes subtly wrong if parameters are set so the data is inverted.

2.6.3 Scope for Improvements

Apart from removing the above noted limitations and deficiencies where that is possible, a number of improvements would be desirable if the system were to be used to any appreciably greater extent.

The considerable range of facilities provided in the plotting system but only accessible by complex and obscure command sequences should be made properly usable by simple and obvious instructions. The existing plotting commands should be made simpler to use by eliminating the need to repeatedly enter unaltered parameters, automatically incrementing the frame serial number, automatically introducing spacing frames in the output, and similar refinements.

Ease of use would also be increased by employing the help facility built in the command interpreter but not used. This requires no program changes, only additions to the generator program input scripts. Error handling also needs improvement; while basically present it has not been fully or consistently implemented.

Improving the file format to include an image size header would also be desirable. Defining the number of image files handled and adding validity checks before accessing any file by number would make the file system more foolproof. Adding routines to associate named files with Fortran unit numbers would be possible using available Fortran extensions (or in Fortran 77) and would remove the present limitation to a small (typically 8) predefined number of files limited by the memory required for their buffers, which is an undesirably arbitrary limit and sometimes inconvenient.

A worthwhile improvement in the plotting speed would be achieved by replacing a number of calls to standard graphics routines from the plotting

program by a single call to a special routine to write all the data needed to plot a pixel onto the file for the microfilm plotter. Even without using assembler a two fold gain in speed should be achieved. The quality of the plot would be improved by switching the plotter to its finest step size, which can similarly be achieved by circumventing the standard graphics package.

A change not in the plot program itself but in the way it is run would allow images of more than 220 by 220 pixels to be plotted. This limit comes from the limit on length of file that will be sent for plotting. If the program were to be run as a batch job remotely at the University of London computer centre, or the plot file sent there on a magnetic tape by courier, this limit would be removed. If combined with the speed and resolution improvements suggested it would be possible and tolerably fast (about 80sec processor time) to plot 512 by 512 pixel images.

2.6.4 Non-standard Language Features

The Fortran IV language used incorporated non-standard extensions. Those used are detailed here to facilitate conversion of the programs to run on other systems.

a) block if. The if...then...elseif...else...endif construct was used throughout the system.
b) free use of expressions. Expressions were used in do loop limits, output lists, and as subscripts.
c) quoted literals. These were used in most cases in preference to the Hollerith form.
d) free format I/O. This was used frequently in minor ways, mostly to output messages to the terminal.
e) end parameter in read statement. This was always used as the means of detecting end of file on input.

All these are standard features in Fortran 77. Conversion to standard Fortran IV would be simple and mechanical, except for the literals in the generated part of the command interpreter, but the lack of a block if would make the programs hard to read.

f) shift and bit-by-bit logical operations. Used for pseudorandom number generation, and character manipulation in the command interpreter and generator.

These are inherently machine dependent. Fortran 77 has adequate character handling, and suitable random number functions are usually provided though not standard.

g) seconds function returning the processor time used. Use to report and limit usage of time by the plotting routines, or report on request from the command interpreter.

A non-standard function that would have to be replaced by an equivalent if available, or the capability deleted.

h) octal constants, the characters ' and ? in literals, do loops in data statements, one use of encode/decode, and occasional mixed real/integer expressions.

These are non-standard but trivial.

i) many routines assume that one or two of their variables retain their values from one call to the next.

This is specified neither by the standard nor by the reference manual for this Fortran. It is however virtually always true. To be strictly correct these variables would have to be placed in common blocks.

2.6.5 Choice of Language

The extended Fortran IV language used proved adequate for the task: Fortran 77 would be equally good, and would avoid the machine dependence created by a non-standard language. Standard Fortran IV would have made the job very difficult by its lack of any if...then...else construct.

I would however be very reluctant to use Fortran for any job of comparable size and character. The restriction on naming variables and the consequent inability to use adequately mnemonic names, while merely irritating in small programs, made writing some of the long subroutines needlessly trying. Altering them later was much worse. The lack of dynamic memory allocation several times forced clumsy use of large fixed arrays allocated by the program to different uses. The lack of both an adequate hierarchical structure and a macro preprocessor in the compiler (as in C) made it impossibly tedious to keep lists of variables shared between routines, in Fortran necessarily by common blocks, simultaneously updated, and forced the use of annonymous, overly long, arrays instead, that could only be understood by looking up a table listing their contents.

This is a list of some of the worst annoyances, but not all. The use of any more modern general purpose language would avoid at least some of them and would be preferable. It is accepted that no language is ideal, and that to some extent it is a matter of individual taste, but algol-68, C, or Modula-2 would have been much better than Fortran: my preference would in future be for Modula-2.

Any choice of language should depend not only on the language and the availability of any special packages required, but also on the quality of the compiler available. Speed, thoroughness of error checking, and helpfulness in reporting errors are vital, and though Fortran was by no means an ideal language, its MNF5 compiler was very good indeed.

All the languages considered above are conventional, compiled, languages. No compiled language would have been ideal for this work. Most of the command interpreter was required merely to do what any good interpreted language does: execute a routine when its name was typed in. The main disadvantage though was in developing the programs: when an error was found the program had to be stopped, the program corrected, and then recompiled before it could be rerun. This is much slower than with an interpreter, that should allow a change to be made and execution continued at once: typically this might be done in 2 minutes as against 10 with a compiler. Indeed making changes is so easy with an interpreter that the user of the finished system can easily create special procedures whenever he needs them. They might be just to execute frequently repeated sequences of standard procedures, or to add entirely new functions to the system. An image processing system based on such an interpreted language would be a major improvement.

Unfortunately there are difficulties. The only really powerful language of the sort that is at all common is APL. And in common with most interpreted languages it has the reputation of being slow: how slow depends on the implementation and it might be acceptable. In any image processing system there are just a few small

pieces of code that virtually determine the overall speed, and these really need to be written directly in assembly code. It should be possible to do this even in APL and then the system performance would be very good indeed; but it would be entirely non-standard and need a great deal of specialised knowledge.

There is a second common interpreted language of the same general type but much less power, and that is Forth. It would be more work writing the system than in some languages, but perhaps not unacceptably so, and it is easy to include the key parts written in assembly language. The resulting system should be as fast as in any other language and have the extra power given by easy extensibility by the user. Forth is well suited to small computers and would be my choice for implementing an image processing system on any machine smaller than a VAX.

2.6.6 User Impressions

The system proved very satisfactory in all its major uses. The only functional deficiency was the omission of a directly calculated Hankel transform from the program for one dimensional data, and the only part that was notably operationally inconvenient was the graphics capability added to the same program.

The command interpreter was the major factor contributing to the system's ease of use. It gave the system a single, simple, style of user interface. It allowed a command to be entered in its basic form with only the essential parameters, or with extra parameters to select more elaborate options. It provided adequate protection against mis-typed commands. And the ability to read commands from a file and then have input revert to the terminal was invaluable, allowing complex sequences of commands to be written and then used repeatedly. This was used for example to build standard images for subsequent modification, to set up standard filters for one dimensional deconvolution, and to produce complex graphics.

The command interpreter, along with its generator program, was of at least as much use during program development. New or modified routines could easily be incorporated in a complete program, and then tested interactively. If necessary extra routines to drive the one under test, or to display its output, could be quickly added, and then deleted when testing was complete.

The only real inconvenience of the system was the need for evening operation, forced by the amount of memory it required. While started as a matter of necessity only, the much faster response of the computer when lightly loaded made it strongly preferable, quite apart from the lower cost. It is accepted that although a matter of no consequence personally, it would be intolerable to most users. The response, even then, was much slower than would be expected from a dedicated system: it was often a few seconds for even trivial operations.

3.1 Peaks in One Dimension

3.1.1 Introduction

The first and simplest case to be considered of the observation of a signal in the presence of noise is a gaussian peak in a one dimensional image. This approximates to the observation of an isolated line in a spectrum, either directly scanned or reduced to one dimensional form from a two dimensional image.

The data is assumed to be regularly sampled at a sufficiently small interval, which is taken to be unit length. The noise is assumed to have a gaussian distribution and to be uncorrelated between samples. Mostly it is assumed to have the same amplitude for all samples irrespective of the level of the signal; that is, it is uniform additive noise and the system is linear.

The peak amplitude of the unfiltered signal will be As and the rms amplitude of the noise on a sample An. The width of the signal peak, given as the sigma of the gaussian, will be Ws, and of a gaussian filter Wf. Unless otherwise qualified, signal to noise ratio Sn will refer to the ratio of the peak signal level to the rms noise, after filtering with a matched filter (and hence the best value obtainable).

The rms noise level after a gaussian filter is $An/(1.77Wf)^{0.5}$. It is apparent that the unfiltered noise level is the same as would be produced from equivalent unsampled noise by a filter of $Wf=0.56$. And the unfiltered but sampled noise can for many purposes be treated as if it was filtered with Wf about 0.5.

Simple theory predicts the performance when a gaussian filter is applied to the noisy data.

a) Sn is $1.33Ws^{0.5}(As/An)$.
b) If the filter is not matched, that is, if $Wf \neq Ws$, the signal to noise ratio is reduced by a factor of $(0.5(Ws/Wf+Wf/Ws))^{0.5}$.
c) The position of the signal is indicated by the position of the peak of the filtered data with an rms error for a matched filter of $1.41Ws/Sn$ provided the signal is well above the detection threshold.
d) If the filter is not matched, the error is increased by a factor of $(0.5(Ws/Wf+Wf/Ws))^{1.5}$ provided the error remains much less than Wf.

e) The noise introduces a random error into the peak value of the filtered data of about one part in the actual Sn, provided sufficient filtering is used that the condition in d) holds. (At an actual Sn of 10, Wf must be greater than about Ws/5, or at Sn of 100, Ws/50).
f) A positive systematic error is introduced into the peak value, that reduces as the square of Sn, provided that the condition in d) holds. In this case it is negligible in practice, being substantially less than the random error.

The operation of the routines used was tested and filtered signal and noise levels and their ratios were validated against theory as a preliminary check.

3.1.2 Graphical Presentation

The natural way to present a one dimensional image is graphically. Fig.3.1a shows the appearance produced by peaks of signal to noise from 2 to 32 with different degrees of filtering. The data in this example has been selected so that the peaks show with typical prominence. At low Sn, the peak value of signal plus noise is subject to a large random variation. This is illustrated in Fig.3.1b for Sn=4. Of 25 peaks, about 4 are lost in the noise, while a few are quite prominent.

This variability is obviously to be expected, but its extent can be subjectively surprising. Consider the case of a matched filter and a detection threshold of 2.5 times the rms level of the filtered noise (which gives about a 10% probability of a false, noise, peak in a length of 10 times the width at half height of the original signal peak). Then we expect that:

a) 10% of peaks with Sn 1.2 will be detected, and 10% of peaks with Sn 3.8 will be missed.
b) A peak of Sn 5.6 will be detected in 99.9% of cases, but will appear with a 5% probability to have Sn less than 4.0, and with the same probability greater than 7.2.

It is evident that the apparent strength of weak peaks can be very misleading, even when they are well above threshold.

The different filters used in the example considerably modify the appearance, but the effect on the visibility of the peaks is rather slight. Even with a filter of a quarter of optimum width the loss is very small. The actual Sn will be reduced by a factor of 1.46, but the eye can to some extent discriminate against the higher frequency noise components and the effective degradation is about half this. With no filtering at all (Ws=10), giving a Sn worse by a further factor of 2, the eye rejects the extra high frequency noise and there is little further deterioration.

Clearly a filter mismatch by a factor of 2 will usually be tolerable, and the slight loss in detection efficiency will be masked by the statistics unless the number of peaks is large. Thus a filter half the width of the peaks to be detected can be used, with the advantage that the loss in resolution is only 11%, against 29% for a matched filter, and it can be a little narrower if resolution is critical. Some degree of filtering is beneficial if the data is not coarsely sampled (Ws is greater than about 2), not least because the filtered data looks cleaner and is more easily interpreted.

Nor is it necessary that the shape of the filter be closely matched. The sensitivity to shape is so slight that a uniformly weighted (rectangular) filter of correct width gave no perceptible loss. In practice this would be usable, though a

smooth profile would be better as it would not leave an obtrusive residue of high frequency noise.

The acceptable degree of filter mismatch is also of interest when the width of a possible signal peak is unknown. A single filter will cover a range of signal peak widths of 6:1 with a drop in Sn by a factor of no more than 1.20; and this will be extended to about 10:1 by visual discrimination. Examining data raw and processed through just two filters will detect peaks of any width from a Ws of less than 1 to more than 200 with a sensitivity of over 80% of optimum.

3.1.3 Performance

The detection threshold was assessed under different conditions by examining a large number (over 600) of peaks drawn on the graphics terminal. Even with this number the statistics were not as good as were desirable, but the system was too slow to permit the order of magnitude increase needed. There is inevitably a considerable subjective element in deciding if, and how certainly, a peak is detected: this was minimised by making each group of comparative tests as a continuous run, and by starting a run with dummy tests for practice. The results overall agreed with theory.

The matched filter detection thresholds found, for a length of data of 20 Ws, were 1.9, 2.6, and 3.3 times the rms noise level to give detection classed as marginal, reasonable, and very clear. These correspond to probabilities of false noise peaks in that length of data of about 50%, 10%, and 1%. In a length of 80Ws they were found about 0.6 higher: the calculated false peak probabilities are much the same. Errors in these values are probably no more than about 0.2.

When a true reference baseline was not displayed, thresholds were increased by about 0.6. This corresponds to the uncertainty in estimating the baseline from the data on a short distance either side of the peak, the noise level being effectively increased by a factor of 1.2. The base level is apparently estimated with an error about two thirds that on a peak.

The effect of slight mismatch in filter width could not be demonstrated, nor could the 6% loss expected from using an optimum rectangular profile filter (length 2.5Ws). With gaussian filters 4 and 8 times optimum width the expected deterioration (by 1.46, 2.02) was shown. Narrower filters of ¼, ⅛ optimum width gave demonstrably less than these deteriorations, factors of about 1.3 being found, and with no filter (Ws=10) 1.35: these are uncertain by about 0.1, and the direct comparison of the same data differently filtered is probably slightly more reliable in suggesting 1.2 at ¼.

The accuracy with which peaks could be located was tested in a similar way, setting the terminal crosswire cursor onto the estimated position of the peak. Results confirmed theory, the error being inversely proportional to Sn and optimum for a matched filter, with the position of the peak in the filtered data being the best estimate of the true position. This held from a just detectable peak, Sn=4, predicted rms error 0.35Ws, to at least Sn=32. The residual error with negligible noise, inherent in setting and reading the cursor, was only 0.001Ws. Note that even for a peak that is only just detectable its position can be estimated with an rms error less than 0.5Ws, or 20% of the full width, half height of the unfiltered peak.

When a filter of double the width was used there was a slight loss of accuracy. With a narrower filter, there was a similar loss but rather less. It was necessary to position the cursor to symmetrically bisect the peak rather than just set it on the top, but the loss of accuracy even at ¼ optimum width was then very slight, whereas a factor of 3.1 is predicted for setting on the peak. Even with no filter (Ws=10) the error was increased by only 20%. This applied to at least Sn=32. Clearly filtering is not critical in determining positions, the eye doing a good job of finding the centre of an unfiltered peak. It is a somewhat subjective process, and becomes difficult to do accurately at high Sn. A matched filter or one only a little different in width makes it much more certain as well as a little more accurate.

The filter, even if not of exactly the same shape as the peak, must be smooth. A rectangular filter, which can give nearly as good a signal to noise, is unsuitable, as it leaves a residue of high frequency noise that makes the maximum of the peak a poor estimate of position.

The error in estimating the amplitude of the peak was also checked. This too was in good agreement with theory for matched filters and wider. Narrower filters or unfiltered (Ws=10) showed no detectable loss at Sn=8 and only a few percent at Sn=32. Further deterioration can be expected at higher Sn, as the process of visually estimating the true peak value becomes more demanding. Again, a suitable filter gains more in convenience that in accuracy.

3.1.4 Continuous Tone Presentation

As an alternative to a graphical plot, one dimensional data can be represented by a continuous tone image of a strip varying in brightness along its length. This is shown in Pl.3.1a for Sn 2.8 to 8 and for various filters. The data was chosen so that the peaks showed with typical prominence.

The detection threshold was seen to be indistinguishable from the same data presented graphically, and to show the same slight dependence on the filter width. There was negligible change for filters from ¼ to twice the optimum width. At 4 times, there was a loss in sensitivity by a factor about 1.4. The accuracy of making position measurements was not assessed as it could not easily be done, and the presentation is inappropriate if amplitudes are to be measured.

This type of plot is thus as efficient as graphical display, and has advantages for some uses. Most notably, it can show a lot of data in a small area. While a graphical plot, if reduced to a small size, becomes illegible and a height of less than 50mm or a peak width at half height of less than 4mm seriously reduces readability, the continuous tone representation, if sharp, need be no more than 4mm broad with a peak width of 0.5mm.

The saving could be used to good advantage when representing high resolution spectra. Even when they have been obtained in or reduced to one dimensional form, presentation in the form of a conventional photographic atlas would be at least a useful complement to the graphical plot . The limitations are obviously the inability to read actual values or to see the shape of line profiles, but it has other advantages besides compactness. It allows the positions of features to be read very easily against a scale along the edge, and it is easier to pick out lines or groups of lines than on a graph: they tend to form patterns that are recognisable despite variations in intensities or background profile.

3.1.5 Lines in a Two Dimensional Image

Photographic spectra usually contain essentially one dimensional information in the form of a two dimensional image. They can be scanned and reduced to one dimension, but traditionally they have been assessed directly and this is still the usual practice. They differ from the case in the previous section in that the noise is two dimensional, appearing as graininess rather than transverse stripes, and the intensity often varies across the width.

When the lines are superimposed on a background (usually in absorption) a limiting factor independent of any noise is that sufficient contrast is needed for them to be visible. This depends on the observer and improves with practice, and also varies somewhat, randomly. The actual depth of lines can be modified by the photographic process; it is the depth in the final image that is here considered. The required contrast was assessed for a positive image with absorption lines, as shown in Pl.3.1b.

It was found that for lines much longer than they are wide, they are clearly visible if the depth is at least 5%. With care they can be just seen at half that depth. For short lines the contrast required was greater. The increase was noticable when the transverse gaussian profile was about 3 times wider than the line; when they were equal width, the depths required had increased to 7% and 5%. Rectangular transverse profiles gave similar results to gaussian with sigma one third of their width.

This is in good agreement with related, slightly more favourable, cases, such as the 1.8% depth required for a bar type target of most favourable size to be resolved (Smith, 1966), and the 1 to 1.5% sensitivity to brightness differences across a sharp boundary in visual photometry (Walsh, 1958). Sensitivity to contrasts as good as 0.3% is sometimes stated, but this is only under ideal conditions, a known target, and a low detection probability.

Tests were made of bright lines with additive noise. The lines had a gaussian profile along their length, 3 or 10 times the width of the line. Circular gaussian filters width 0.5, 1 and 2 Ws, elliptical 0.5 by 1.5 and 1 by 3 Ws, and gaussian width Ws by rectangular profile length 25Ws were used, as well as no filter (Ws=1.4) which is roughly equivalent to a filter width 0.3Ws. The theoretical matched filter Sn was calculated as $1.77(WtWs)^{0.5}(As/An)$ where Wt, Ws are the two sigmas describing the lines, As the peak signal value, and An the noise on a sample. The Sn is reduced by mismatch, the factor for one dimension applying for the mismatch on each axis.

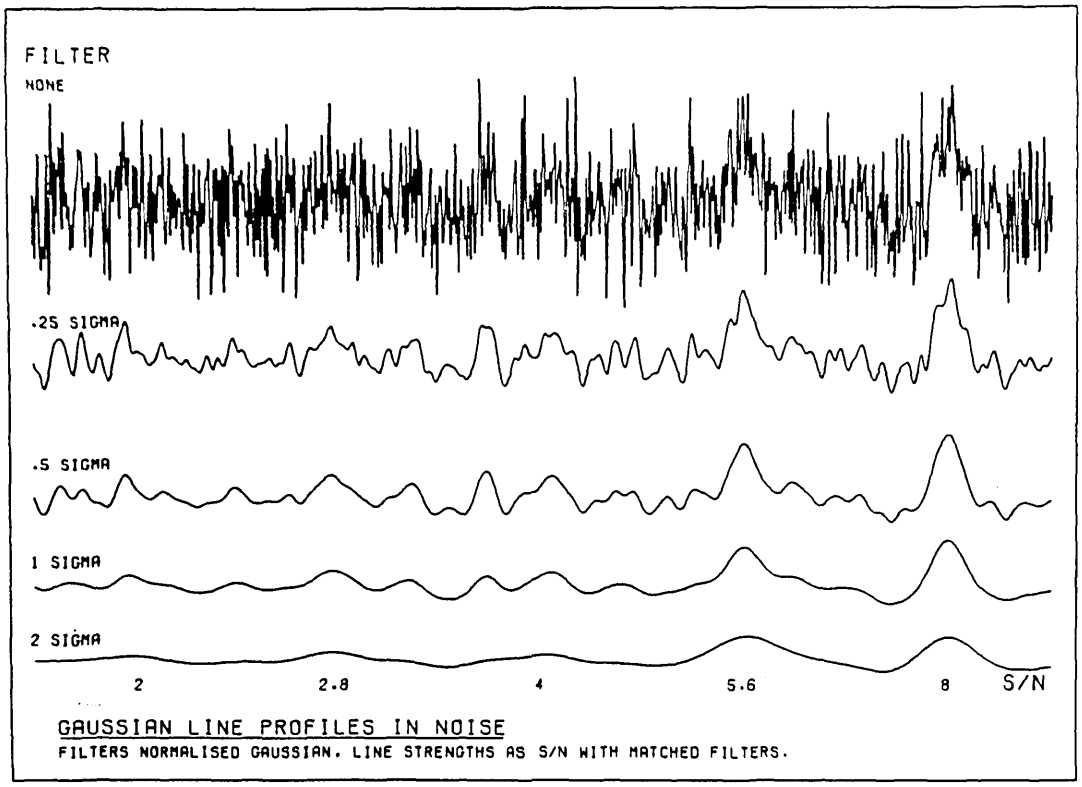
Detection threshold was found to be at Sn about 3, which was not significantly different from the one dimensional results. It was slightly worse for the longer line than the shorter, a difference of about 25%, but this was only marginally significant. Surprisingly any differences between filters were less than this (the very long filter not being applied to the narrower spectra) and were not detectable. The non-circular filters, despite a better match, showed no advantage over the small, circularly symmetric ones or even over no filtering at all. It was apparent when examining the images that the eye was very effective at discerning the lines among noise peaks of different shape, and that when any asymmetric filter gave similarly elongated noise peaks this discrimiation was lost. The results for different filters are shown in Pl.3.1c.

Thus filtering is not really required, except a little if the data is very finely sampled. Performance is little worse than the ideal, provided the lines are not too long. A length of about 10 times the width would seem about the limit at which efficiency begins to drop noticeably, and beyond which processing is required, most simply a reduction to one dimension.

Similar tests were made with dark lines on a background continuum with a gaussian transverse profile. The noise was made proportional to the square root of the signal level, so as to approximate to photon noise. The optimum filtering in this case is matched to the line profile but integrating with uniform weight across the spectrum, which was approximated by a rectangular profile of length 4 sigma. Spectrum widths of 1, 3, and 10 times the line widths were used.

The results were essentially identical to those for bright lines. The detection threshold was again about 3. Filtering had similarly slight effects for the same reasons, except that the finer sampling used made the unfiltered images a little less satisfactory, especially with the widest spectra. Typical images are shown in Pl.3.1d.

To summarise, it is found that the visual detection of lines in two dimensional images does not fall seriously short of optimal. Spatial filtering cannot significantly improve the performance: it may be used, but is not needed unless the data is very finely sampled or the lines are very long. If used, a filter half the width of the line profile is adequate and will avoid undue loss of resolution and contrast. The width can be the same transversely, or comparable to the width of the spectrum, giving results of different appearance but little difference in detection efficiency. For very long lines, more than about 10 times their width, some loss of performance seems inevitable unless the filter has a similar size, in which case the central part of the image effectively represents the data reduced to one dimension. Both performance and recommended processing seem quite insensitive to the type of the spectrum, lines, and noise.



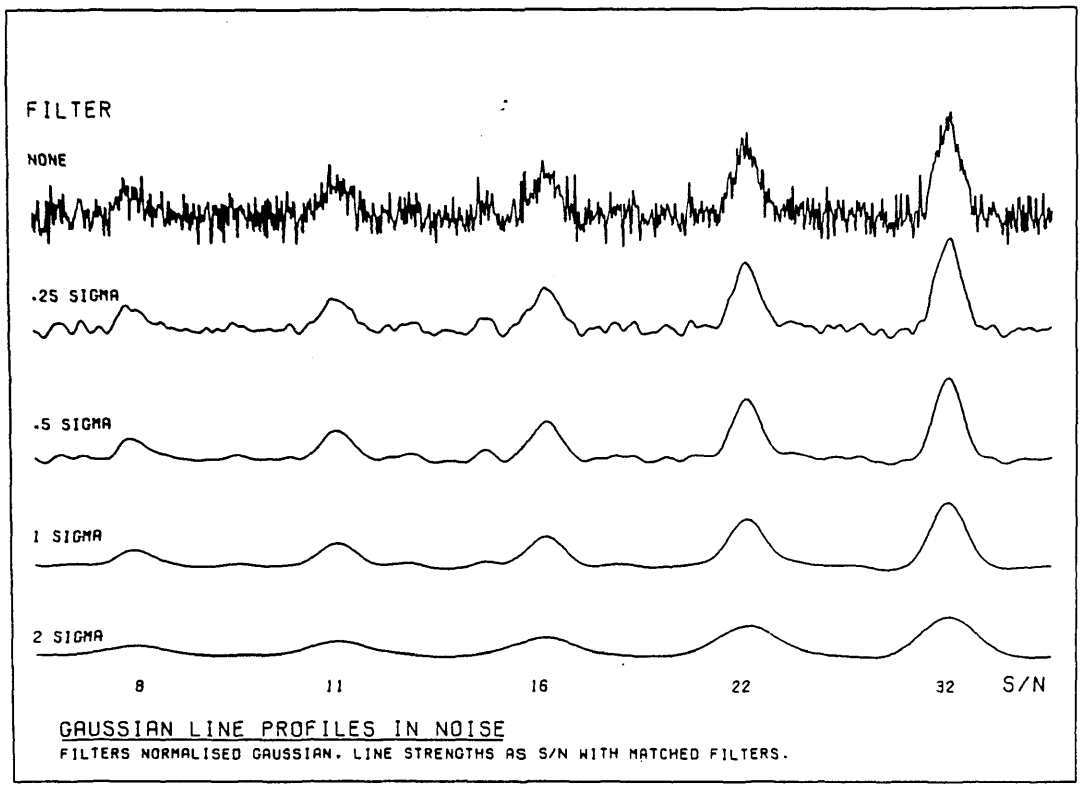


Fig.3.1a One dimensional peaks in noise.

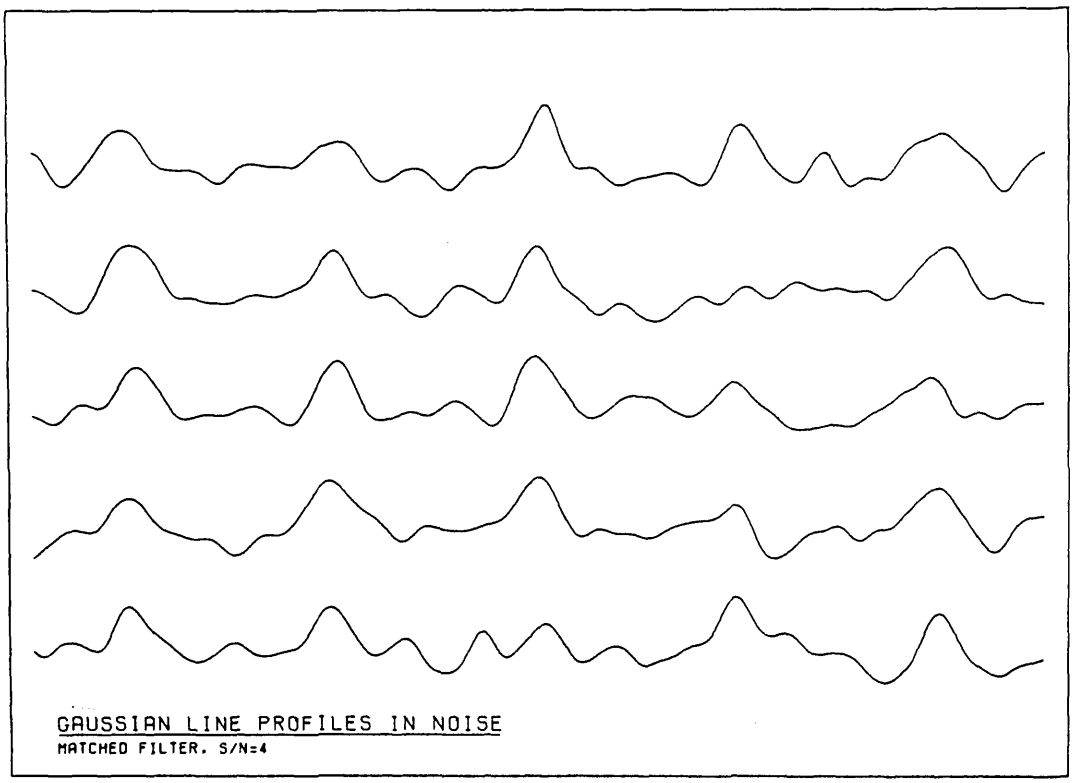


Fig.3.1b Variation in appearance of weak lines.
Twenty-five peaks of equal strength on a regular grid.

3.2 Peaks in Two Dimensions

3.2.1 Introduction

Circularly symmetric gaussian peaks in noisy, two dimensional images form the next case, closely related to the last. The physical interpretation is as star images superimposed on sky background. Only noise with a gaussian distribution independent of the signal level will be considered, corresponding to the case of faint stars with the noise coming predominantly from the background. It also approximates to stars weakly recorded on a photographic plate, the noise coming from the slight background density of chemical fog.

The symbols used will be the same. The noise level after a gaussian filter is now $An/3.54Wf$, and the peak signal level $As/(1+(Wf/Ws)^2)$. Theory gives:

a) Sn is $1.77Ws(As/An)$.
b) For an unmatched filter the signal to noise ratio is degraded by a factor of $0.5(Ws/Wf+Wf/Ws)$.
c) For a uniformly weighted square filter, the optimum size is a side of $2.8Ws$ giving a signal to noise degraded by only 1.12, and varying with mismatch in a way similar to b).
d) Accuracy of position (on one axis) or amplitude, for a gaussian filter, is the same as for the one dimensional case.
e) The position error is increased for a mismatched filter by $(0.5(Ws/Wf+Wf/Ws))^2$.

Similar checks were made on the working of the routines to be used.

3.2.2 Presentation of Images

The simplest way of representing an image, and the way closest to its original form, is as an image, continuous toned and linear. The appearance of a series of stars is shown in Pl.3.2a both unfiltered and after a matched filter. The effect of different degrees of filtering is seen in more detail in Pl.3.2b. It is clear that a little filtering improves clarity, but a matched filter is more than is needed and incurs an unnecessary loss of resolution. The appearance of the noise changes with filtering, an overall graininess being turned into a mottling. For $Wf/Ws=½$, the mottling is

substantially finer than the star images, while for a matched filter there is less difference and clarity is less even though signal to noise should be 25% better.

The square filter gives a very similar appearance to the gaussian, with just a little more high frequency noise, which shows a slight streakiness oriented along the edges of the filter. The stars show no visible tendency to squareness even with the filter large enough for optimum signal to noise, and the distortion would be even less for a, generally preferable, smaller filter. There is, qualitatively, little advantage in using the slower and more elaborate gaussian filter.

It is a simple modification to limit the plotted data to above a chosen minimum. Correctly set, this acts as a detection threshold, the noisy background is eliminated, and no subjective judgement is required to decide if a peak is significant or just noise. It is illustrated in Pl.3.2c.

There are two drawbacks to this, shared to some extent by even the more elaborate non-linear methods such as maximum entropy fitting. The worst is that one is entirely dependent on the correct setting of the threshold, and it is impossible to tell a faint star image from a noise peak passed by incorrect discrimination. In contrast, with a linear image there is all the necessary information to judge the significance of a peak, and one only marginally above the noise is clearly seen to be doubtful. The lesser is the distortion of the relative brightnesses of stars, stars only just above the threshold appearing much too weak. Examples are given in Pl.3.2d.

The first defect seems sufficient to prohibit in most cases the use of this method except to complement a linear image. The threshold can be set in simple cases from the measured distribution of data values, but for best results trial and error adjustment may be needed. Its main practical limitation is that a single threshold across the entire image does not allow for any unevenness of background. More complex methods can avoid this.

Contour (or isophotal) maps are more appropriate for quantitative rather than qualitative assessment, and while a cutoff produced by setting the lowest contour level above the noise is needed to keep the map reasonably uncluttered the above mentioned disadvantages are less serious for such use. They are often used when continuous tone images would be more suitable, because of restricted computer output facilities. This is regretable as it makes it much harder to take in a complex image, and it is difficult to tell if slight irregularities in contours are significant or merely noise.

The general appearance of a contour plot of a series of stars is shown in Fig.3.2a, which can be compared with the same as continuous tone images in Pl.3.2a. It is evident that filtering is essential if the map is to look reasonably clean and detection sensitivity not be needlessly sacrificed.

Contours need not be linearly spaced. If the interval is not to be varied arbitrarily, a logarithmic spacing is most natural, and is well suited to the astronomical use of magnitudes. The appearance of a star is not greatly different in the two cases, the linear spaced contours getting wider spaced at the lowest levels, and the logarithmic continuing to get closer. See Fig.3.2b.

Logarithmic spacing is generally more useful, satisfactorily plotting a wide range of data, but for this reason not showing relative intensities very prominently. In the examples the spacing is ¼ magnitude, which is too close: ½ magnitude answers best for most purposes. Linear spacing is necessary if the background is

uneven, as otherwise variation from the assumed level will seriously distort appearances.

The effect of different filtering is shown in Fig.3.2c. Less than matched filtering is clearly sufficient, Wf/Ws=½ seeming reasonable and not sacrificing too much resolution. That much is needed if sensitivity is not to be lost unduly or the plot cluttered with a vast number of tiny contours round noise peaks. A square filter is not suitable, producing very irregular contours.

3.2.3 Noise Peaks

The noise at any point remains after filtering gaussian, so the probability of it exceeding a chosen level is easily calculated. To find the probability of the level being exceeded at any point in an image requires that we know the number of independent noise values that it effectively contains, the filtering having introduced correlation between the noise on different samples. This can be expressed as an area corresponding to one independent value.

The area was found by numeric tests by listing the peaks in 20 images of filtered noise, each of 100 elements a side, Wf=2. It was determined to be $(1.2Wf)^2$, the constant being possibly 10% in error: as expected, it is of the order Wf^2. This was based on peaks from over 2.5 to 4 times the rms value of the noise, about 300 in all, the highest being 4.3. It was confirmed that for a square filter the effective area was about the same as for a gaussian filter giving the same rms noise level (both filters normalised): very roughly a fifth of its geometric area. These results, though approximate, are close enough for practical use.

We can calculate the threshold level corresponding to a given probable number of spurious noise peaks for a specific filter and area of image. A few examples will be given.

a) If a star is being searched for close to a known position, perhaps a close companion of another star or the optical equivalent of a precisely known radio source, we might be interested in any object within 3 arcsec for 1 arcsec seeing. Then for a 10% risk of identifying a false peak, with a matched filter we need a threshold set at 3.0 times the rms level of filtered noise.
b) For an image of a star field of 120 by 120 pixels, with a filter Wf=2, there is a 10% risk of a false peak with the threshold at 4.0: at 3.0 there will be typically 3 or 4 noise peaks.
c) For a square plate covering 20 arcmin with 1 arcsec seeing, with a matched filter a threshold of 5.0 would give an average of one false peak: with 6.0 there would be only a 0.5% probability of one.
d) For the whole sky at 1 arcsec seeing a threshold of 7.5 gives a 10% probability of a false peak.

While thresholds in the range of 3.0 to 7.5 cover almost any conceivable use, 4 to 6 covers all normal requirements, with 4 typically appropriate to small images. The total range corresponds to a one magnitude variation in detection sensitivity over the extreme range of search areas and risks of error, while between searching a small area and the whole of a normal telescope plate (assuming the size of one in arcsec is equal to the other in arcmin) the difference is usually from 0.3 to 0.5 magnitudes.

3.2.4 Detection Efficiency

When the peaks in an image are found numerically, the detection performance simply follows theory. Thus, for example, a field of 16 equal stars in noise, after matched filtering, gave one star above the highest noise peak at Sn=2. Increasing the brightness of the stars gave 8 detected at Sn=4, 13 at Sn=5, and all at Sn=6. Images with a cutoff set as a discrimination threhold and contour plots with the lowest contour level set similarly gave equally ideal performance. Departure from matched filtering will give the full theoretic loss, so Wf/Ws=½ giving 80% of the ideal sensitivity will usually be the minimum tolerable filtering.

The detection of peaks in a continuous tone, linear image is less certain, depending on visual discrimination. It was found that with matched filtering a peak could just be recognised provided it exceeded 3.0 times the rms noise level. At 5 times a peak was easily visible, while at 7 times it was obvious. This sensitivity is more than sufficient for normal use. Even in the small images being used (100 by 100 pixels, Ws=2) the highest noise peak was usually about 4 times the rms level. It was quite easy to compare faint peaks, and judge the appropriate threshold from the rapid increase in number of noise peaks at lower levels. Visual discrimination was found to be just as effective as numeric.

While the basic tests were made with grids of stars in known positions, some were also done with random starfields. These confirmed the results and showed the degree of uncertainty in judging the threshold in a crowded field. To avoid accidentally including noise peaks because of this and errors in comparing brightness it was necessary to be a little cautious: the effective threshold could be up to 0.5 times the rms noise level above the ideal. There is a similar difficulty in setting a numeric threshold, and it can be too high by a similar margin.

The effect of narrower than optimum filters was less on visual discrimination than on numeric. At Wf/Ws=½ there was no detectable loss of sensitivity, and the clarity of the image was much the same, though the appearance was different, the noise peaks being much finer than the stars. The gain in resolution was obvious. Narrower filters produced a less clear image, the grainy appearance tending to conceal the stars, and some loss of sensitivity with only slightly better resolution. This was much less than the predicted drop in signal to noise ratio and even without filtering (Ws=2) the sensitivity was no less than 80% of optimum. An optimal square filter gave a similar performance to the matched gaussian, except that the noise was more obtrusive, having more high spatial frequencies and a small scale streakiness oriented along the edges of the filter.

To summarise, detection is close to theory, whether numeric or visual. For numeric, a filter of nearly matched width is needed, Wf/Ws=0.5 or 0.7 being appropriate according to whether or not maximum resolution is desired at a slight loss of sensitivity. For visual, filtering is not so critical, but Wf/Ws=½ will give nearly optimum performance and an attractive appearance with only a slight loss of resolution.

3.2.5 Accuracy

The predicted error in the measurement of the intensity of a star in the presence of noise was confirmed as being just the rms noise level, and the variation for filters of different width demonstrated, a gaussian within a factor of 2 of

optimum width or an optimal square filter being shown to be generally tolerable. The peak finding routine was shown, despite its inherent approximations, to introduce a generally negligible amplitude error of 1% rms for Ws=Wf=1, and 0.07% at Ws=Wf=2.

When the brightnesses of star images were estimated visually against a series of images (after a matched filter), it was confirmed that at low Sn the accuracy was about as predicted. At Sn greater than about 15 the uncertainty of the visual comparison was dominant, the images being compared not being brought close together. The rms error for bright images was about 7%, or 0.07 magnitude. This is comparable to the 0.05 magnitude error typical of visual variable star observation comparing them to other stars in the field (Glasby, 1968), and the 0.1 magnitude rms error typical of photographic photometry within one plate using visual comparison to a reference series of images (Stock and Williams, 1962).

If the detection threshold is typically taken as 5 times the rms noise, the detection probability of a star will increase from 10% to 90% as its brightness increases from 3.7 to 6.3, a range of 0.6 magnitudes. For a star ½ magnitude above the brightness for a 50% detection probability, the rms error expected is 0.14 magnitude; at 2½ above it is 0.02; while a 0.001 magnitude rms error requires 5.9 above the limiting magnitude. All these assume a linear system with noise independent of signal level.

The theoretical positional accuracy was also confirmed. With a matched filter an rms error of 0.27Ws was found on a single coordinate at Sn=5, and 0.045Ws at Sn=30. The loss with a mismatched filter was as predicted, a width mismatch by a factor of 2 increasing the error by half. A square filter gave as expected lower accuracy. In the absence of noise the peak finding routine produced with a matched filter an rms error of 0.015Ws for Ws=1, 0.002Ws at Ws=2: which for practical purposes are negligible.

Traditional photographic astrometry gives relative position over a small area of a single plate with an rms error of approximately 0.02 arcsec. (See, for example, Strand, 1963). If the seeing is typically such that Ws is 0.8 arcsec we see that a star 2½ magnitudes above the plate limit is theoretically bright enough to attain this accuracy. Even for a star close to the limit positional errors assuming 1.5 arcsec seeing will be only 0.2 arcsec rms.

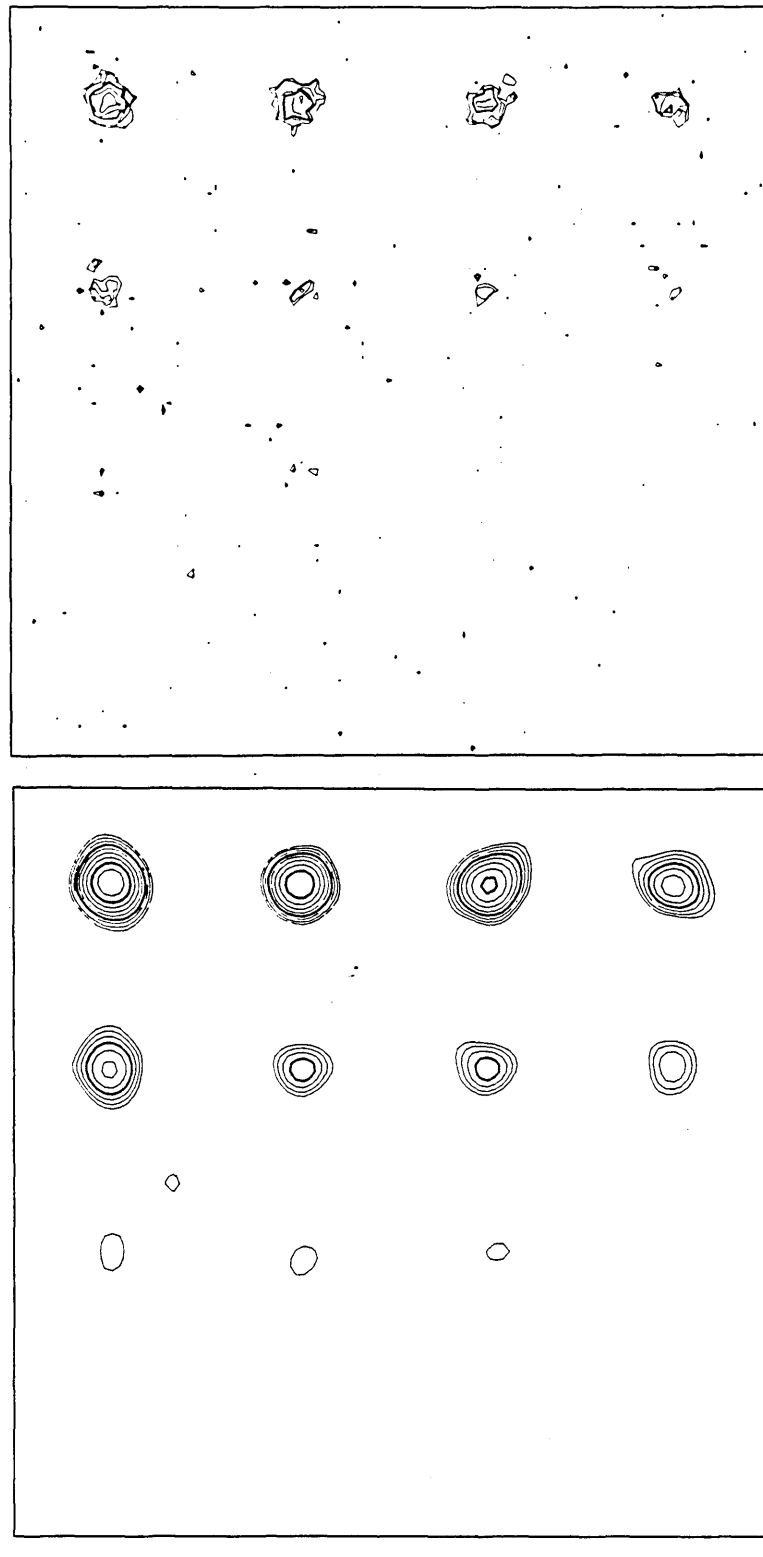


Fig.3.2a Contour plots of stars before and after matched filtering.
Upper: unfiltered. Lower: filtered.
Stars decreasing in steps of ¼ magnitude from a matched filter signal to noise of 32.
These show the interrupted contours produced by limiting the number of contours that will be plotted in a cell. Images are 101 pixels square. Star profile sigma is 2 pixels.

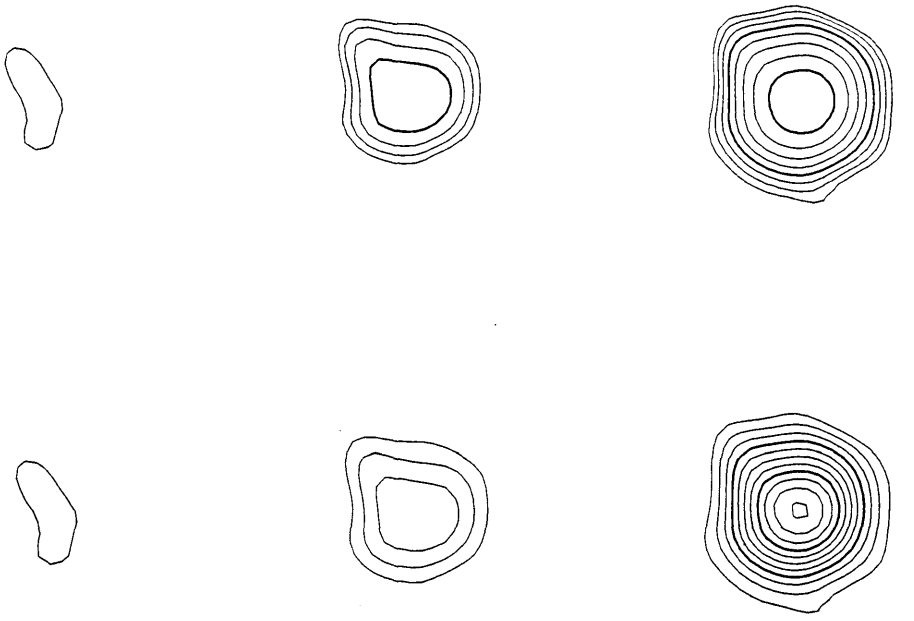


Fig.3.2b Comparison of linear and logarithmic spacing of contours.
Upper: logarithmic, ¼ magnitude spacing. Lower: linear.
Filter: Wf/Ws=½. Matched filter Sn: 5, 12.5, 31.

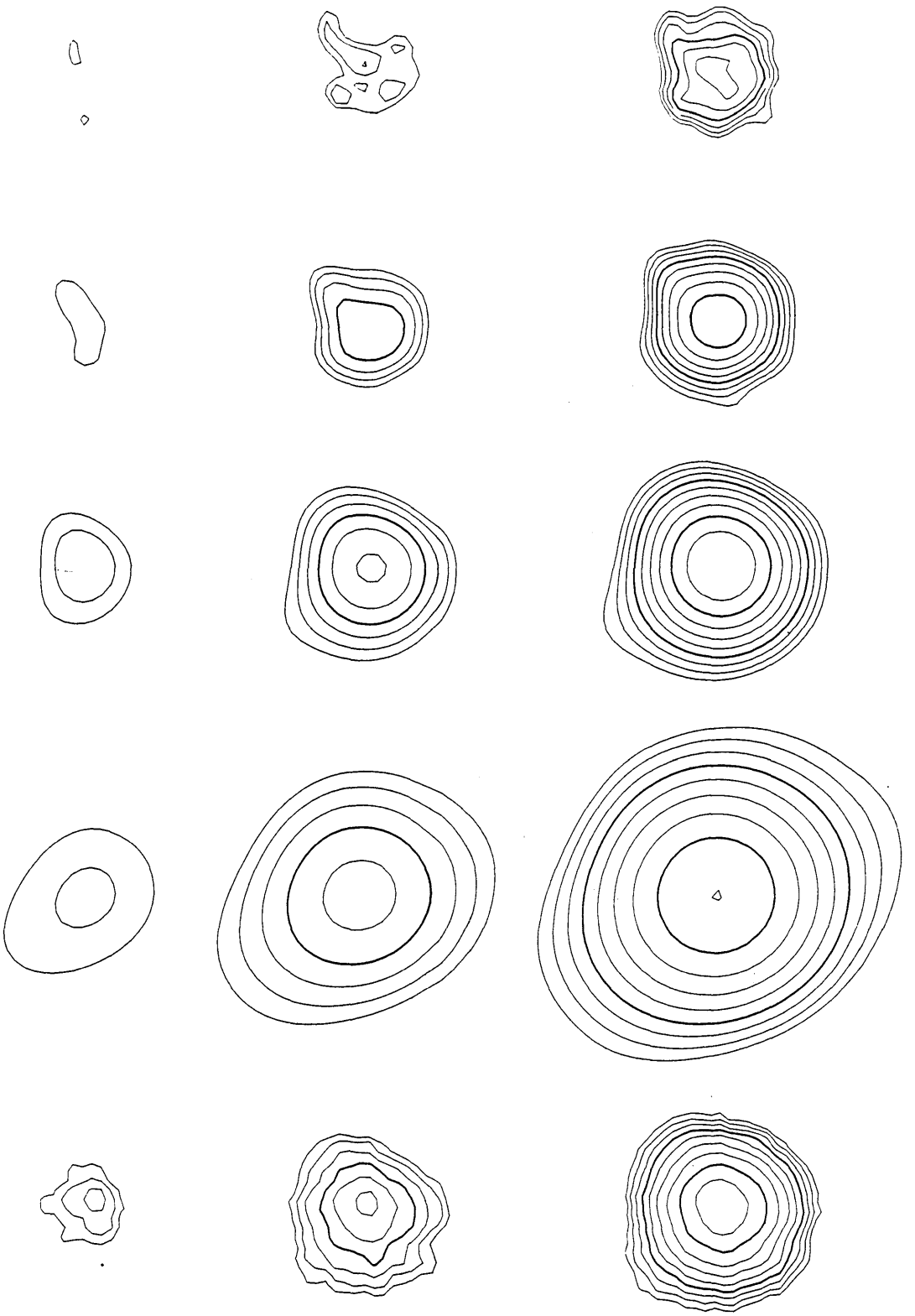


Fig.3.2c Contour plots of stars.
Filter (t to b): gaussian Wf/Ws=¼, ½, 1, 2; square filter side 2.8 Ws.
Matched filter Sn: 5, 12.5, 31. Contoured at ¼ magnitude.

3.3 Resolution in One Dimension

3.3.1 Introduction

The resolution of an image as characterised by the ability to separate closely spaced peaks is an important property. In one dimension it describes the separation of close pairs of spectral lines.

Two criteria for separation were used. The resolution was taken as the least separation at which the peaks were clear enough for their spacing to be measured. The discrimination limit was the least spacing at which the combined peak could be distinguished from a single one. The variation of these with the ratio of component peak amplitudes, noise level, filtering, and type of presentation was investigated.

For a graphical presentation a double peak and noise were generated and filtered separately, then displayed on a graphics terminal summed in different ratios. If possible the crosswire cursor was used to measure the apparent separation and ratio, until the appearance corresponding to resolution into measurable components could be reliably recognised. At ratios over 32 the vertical scale of the plot had to be increased, truncating the top of the profile, or the quality of the display was inadequate to show the full resolution. About 400 profiles were displayed, which gave adequate but not good accuracy: better could have been obtained in reasonable time had the computer system been more suitable for interactive graphics and not had a response time often longer than 5 seconds.

Continuous tone images were plotted on microfilm. Three types were used: one dimensional data and noise displayed as a strip of varying brightness, and two types of two dimensional image with gaussian transverse profiles to the spectra. One had bright lines and uniform noise; the other, dark lines on a continuum and noise dependent on signal level in an approximation to photon noise. About 1000 double lines in 150 spectra were examined.

The results are summarised in Fig.3.3a. The appearance of double lines in graphical output is illustrated in Fig.3.3b, and examples of continuous tone images are given in Pl.3.3a. In the detailed results that follow, the same symbols will be used as in previous sections, and separations will be expressed in units of the line profile sigma Ws.

3.3.2 Graphical Output

In the absence of noise the resolution to give an apparent separation within 10% of the true value and a ratio within 20% was about 2.5 (to within 10%) for ratios up to 10, increasing to 3.4 at 100. It was comparable to the resolution indicated by the zero dip criterion and better than it except for ratios very near one, a weak peak being clearly distinguishable when just a hump on the side of the stronger.

Doubleness was evident to smaller separations, the limit being about 0.8 less at small ratios, and only half that at 100. The orientation of markedly unequal pairs was always evident from the asymmetry that was the first sign of their structure.

In the presence of noise filtering was helpful, less than matched filtering being appropriate to retain nearly the full resolution. It was found that the resolution was not critically dependent on Sn, so a filter can be chosen mainly with a view to sharpness. Generally $Wf/Ws=½$ is suitable, which increases the effective width of the peaks by 12%, but when sharpness is critical $Wf/Ws=¼$ is more appropriate, giving a negligible 3% broadening. In this case the signal to noise ratio is reduced by a factor of 1.46 but the effective loss is less than this. In this work whatever filter was more suitable was used.

Noise produced no noticeable degradation of resolution if the weaker component had Sn greater than 20; that is, was 2 magnitudes above the detection threshold. This was tested at ratios up to 32. For a separation equal to the resolution, a weaker component Sn of 10 was found sufficient to ensure that doubleness would most probably be evident.

The doubleness discrimination limit was found not to be impaired if Sn for the sum of the two peaks was about 300 or more; that is, about 5 magnitudes above detection threshold. This was shown to a ratio of 32. It is substantially less than the effective limit to Sn of at least 1000 set by the limited resolution of the graphics terminal, so it and the discrimination limits in the absence of noise should be fairly reliable. Some dependence on the type of display seems inevitable.

At large separations the two peaks were essentially independent. This was found to be the case at separations of 4 or more at low ratios, increasing to 4.5 at 32.

3.3.3 Continuous Tone Images

Resolution in this case was based solely on the appearance of the separation being measurable, no attempt being made to check the accuracy with which this could be done. In cases where significant filtering was used, the loss in resolution was found to agree with theory, and the loss was corrected before combining these resolutions with others. Moderate filtering made only a slight difference to the effect of noise; when the noise level was expressed relative to the observed detection threshold for the weaker component alone, this and other departures from ideal detection efficiency were of no consequence.

The most notable result was that there was no significant (10% or more) difference between the three types of images used. Nor was there between spectra

of different widths, from 1 to 10 times the line width, which was the entire range used, except that moderate widths seemed easiest to read.

The results were very similar to those for graphical output. Resolution was very slightly better at low ratios (up to about 3), typically 2.0 against 2.3. This is a result of the eye having a better response at medium than low spatial frequencies, giving the appearance of a clear dip even when the profile does not quite show one. At high ratios (over 10) the resolution was worse than for graphical output, 3.3 against 2.9 at a ratio of 27. The loss is due to insufficient contrast sensitivity of the eye: at 27 the weaker line in a dark line image can only just be deep enough to be seen, and even at 10 the weak line is not conspicuous. A bright line image is a little better, but high ratios are still only clearly visible if the lines are well separated and the image is non-linear, the bright component being allowed to saturate.

The doubleness discrimination limit is a little worse at low ratios than for graphical output, slight distortions of line shape being undetectable by eye. It deteriorates badly at high ratios, at which if a line is to be discernably double the weak component has to be seen clearly separate.

Noise has negligible effect on resolution, provided the weaker component exceeds the detection threshold by a factor of about 5, or by 2 magnitudes. This is the same as for graphical output. The discrimination limit is affected, but less than in the graphical case.

3.3.4 Summary

The resolution of peaks in one dimension has been shown to be substantially independent of the type of presentation. The separation needed for clear resolution is approximately 2.2 at low ratios, increasing to 3.0 at a ratio of 30.

This is very similar to the resolution indicated by the zero dip criterion, and justifies its use. It is substantially better than the 2.65 for equal components given by the 20% dip equivalent to the Rayleigh criterion, and the actual 2.0 found to show a clear apparent dip in an image is even more superior.

Resolution is little affected by noise, provided the weaker component is at least 2 magnitudes above the detection threshold. It is reduced by filtering as indicated by theory, but some filtering is desirable to improve both appearance and detection threshold. A filter a quarter of the width of the line maintains essentially full resolution with only a small increase in effective noise level; if noise and not resolution is critical, half width, sacrificing 11% resolution, is more appropriate.

Doubleness can be discerned at separations less than that needed for clear resolution. At best, for low ratios and graphical display, 1.5 is needed. More usually, it is 0.5 or less better than the resolution. Graphical output is the more sensitive, and a total intensity 5 magnitudes above the threshold is needed for best results at ratios up to 30.

Images are slightly inferior to graphical plots in most respects, and are subject to more limitations. Most seriously, at ratios of 10 or more it tends to be difficult to see the weak component, and over 30 virtually impossible if linearity is preserved because of the limited contrast sensitivity and dynamic range of the eye. It is also harder to see the slight distortion of line profile at small separations. Only in the resolution of strong, nearly equal line pairs are they slightly superior. While the width of a spectrum image does not much affect the resolution, 3 to 5 times the line width generally gives the clearest appearance in the presence of noise.

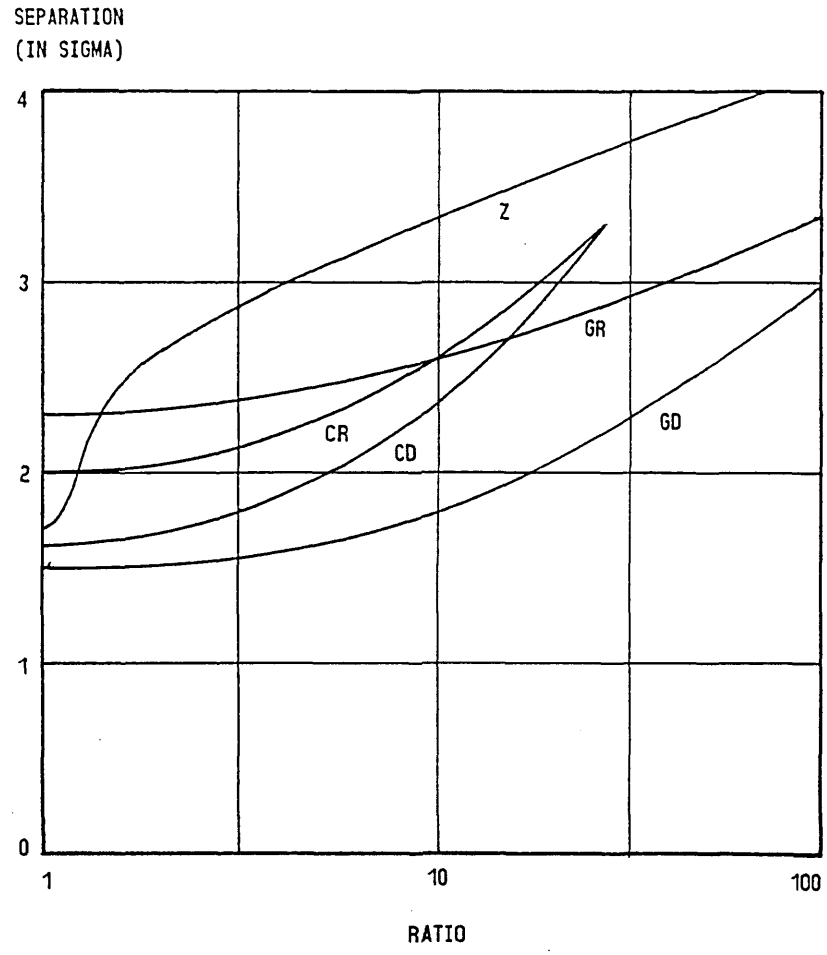
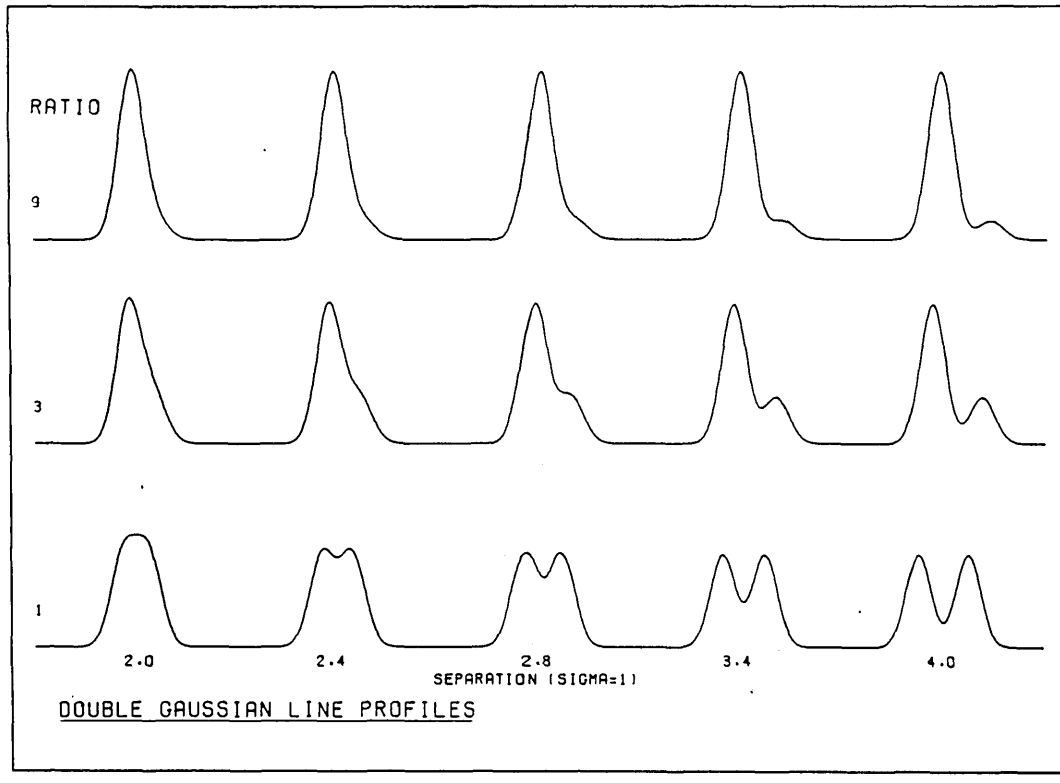


Fig.3.3a Resolution in one dimension: double lines.
Accuracy: errors are probably 0.1 or less in separation (2 sigma).



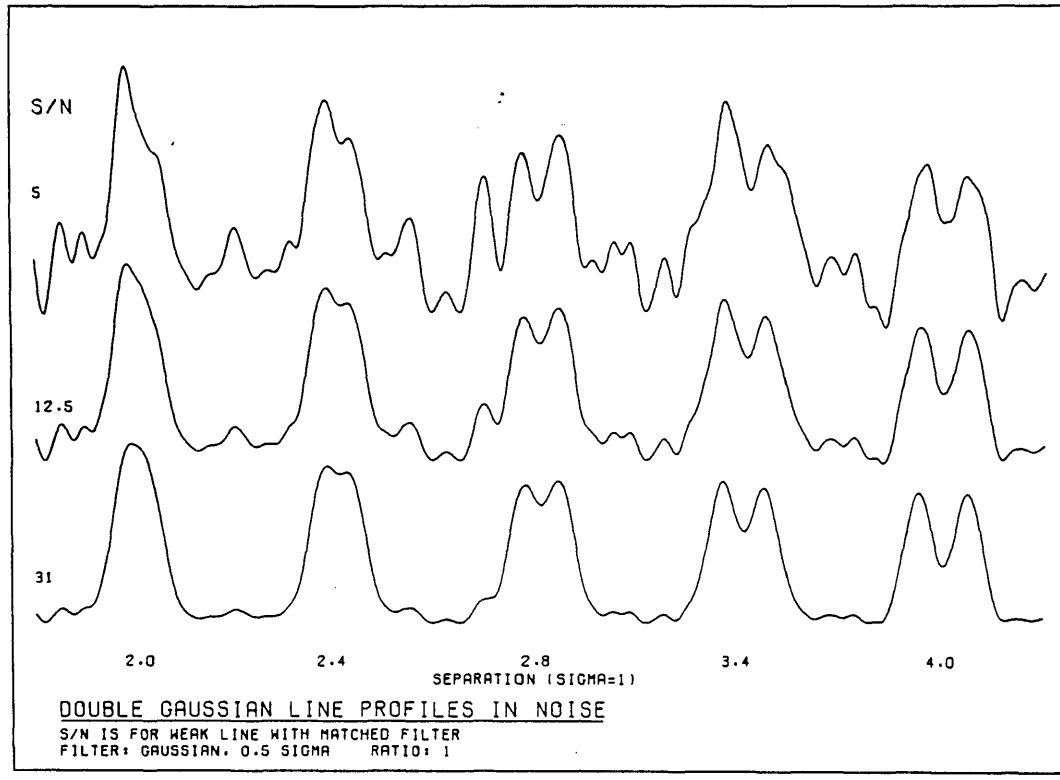
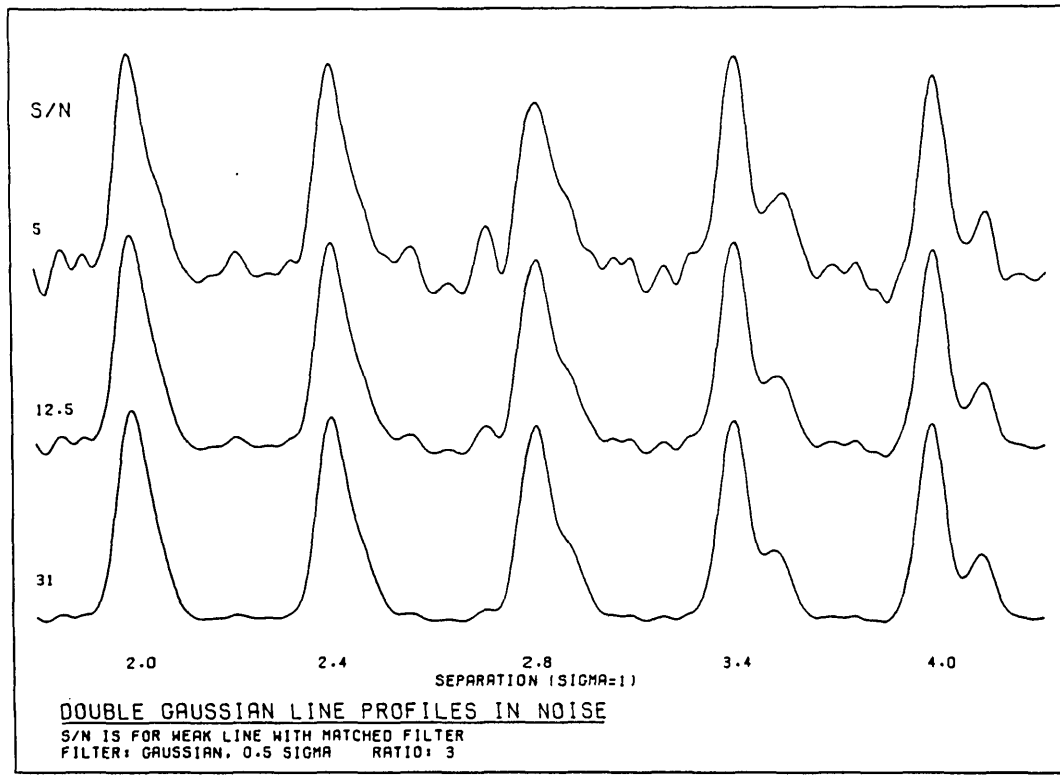


Fig.3.3b(i) Appearance of double lines.



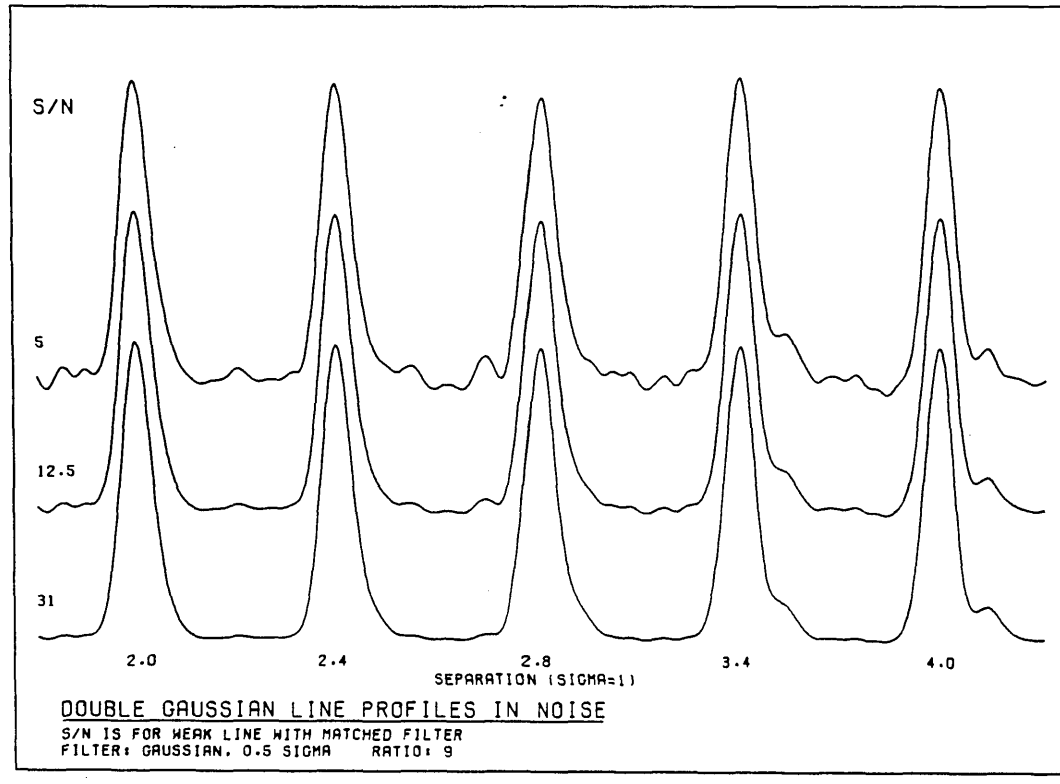
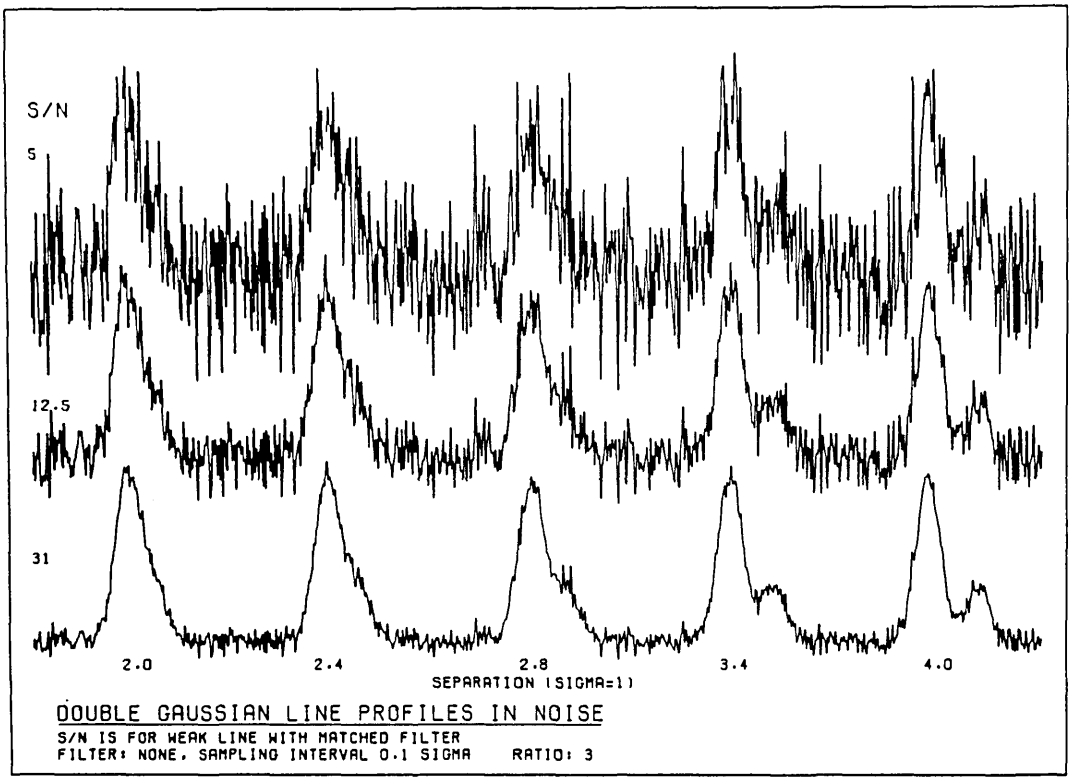


Fig.3.3b(ii) Appearance of double lines (cont).



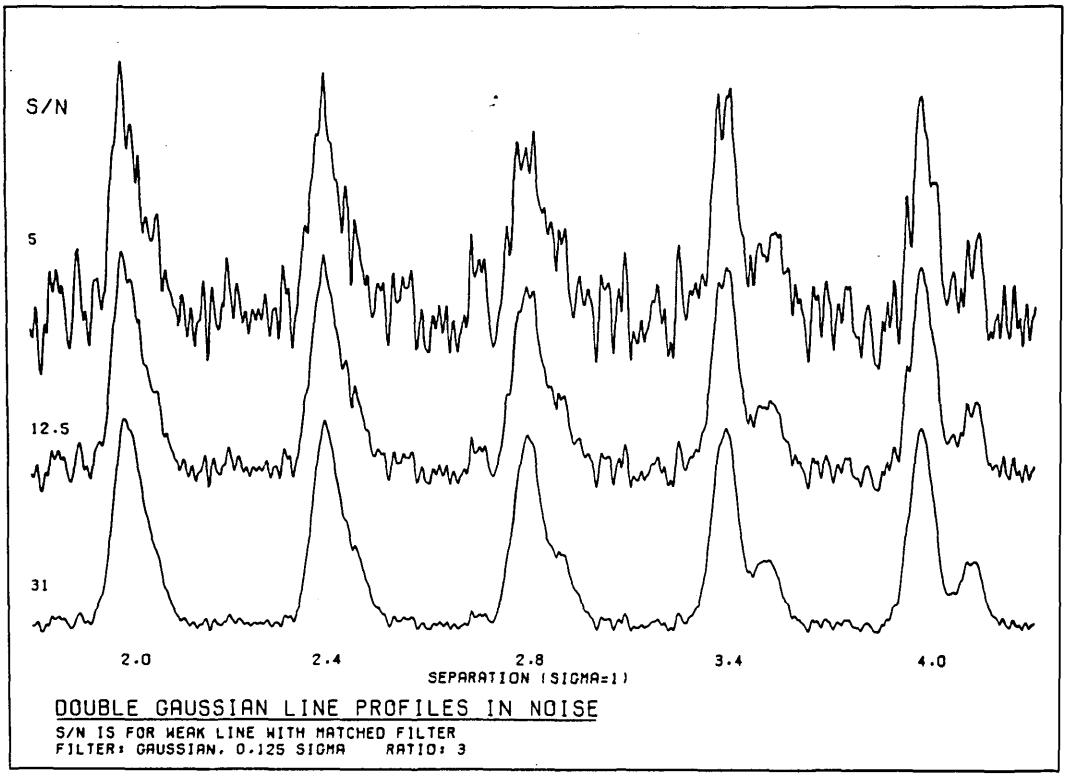
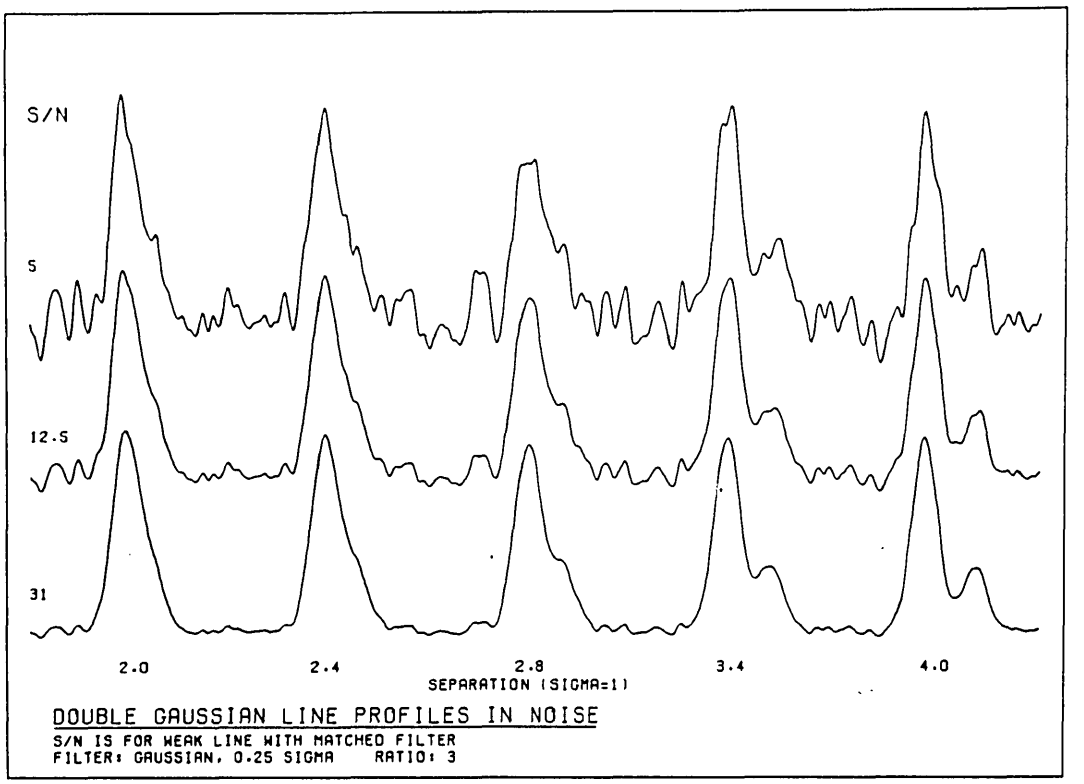


Fig.3.3b(iii) Appearance of double lines (cont).



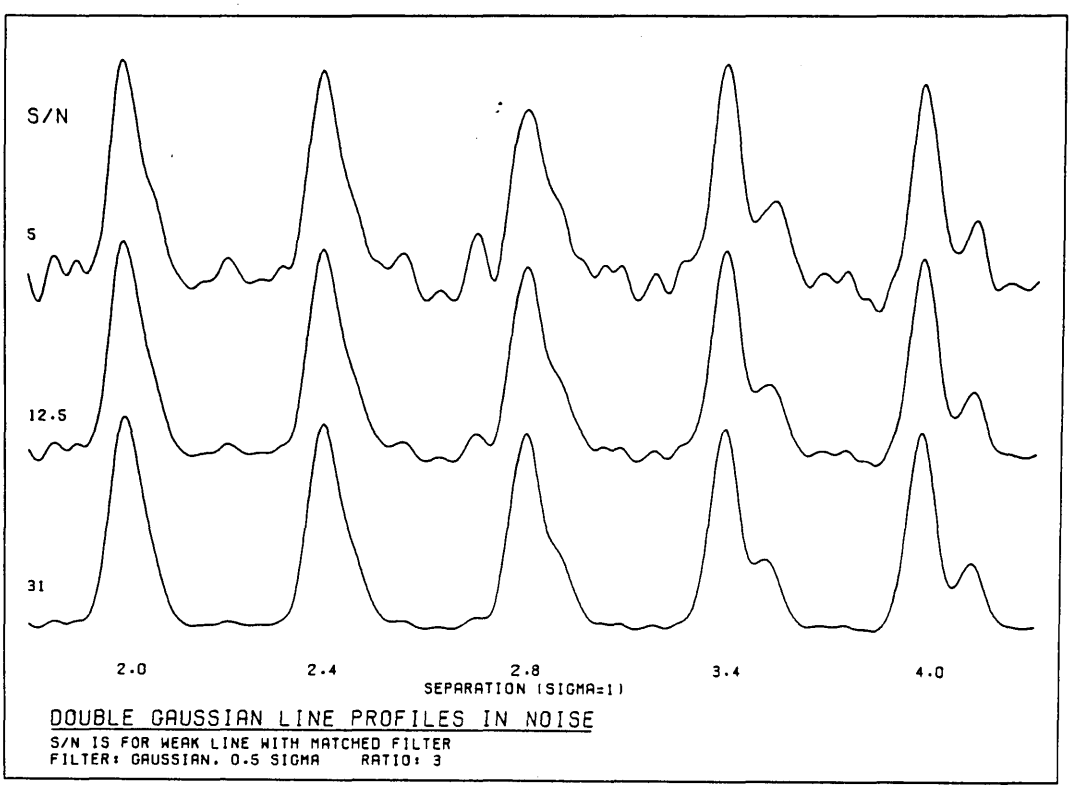


Fig.3.3b(iv) Appearance of double lines (cont).

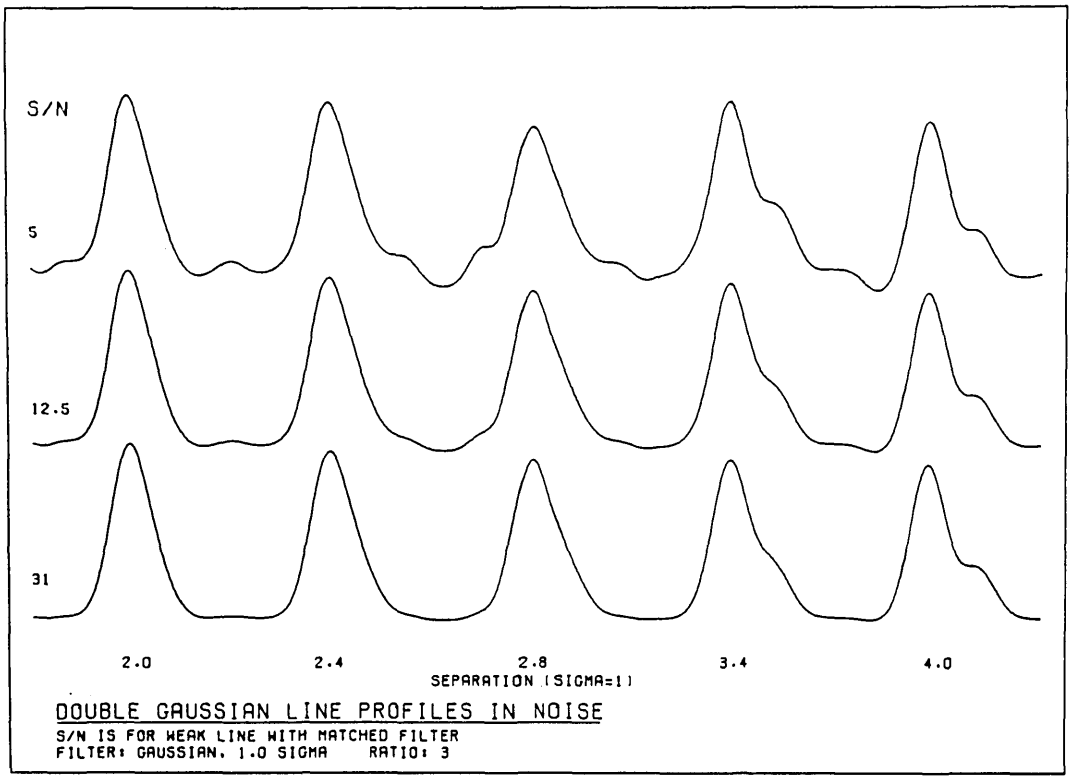


Fig.3.3b(v) Appearance of double lines (cont).

3.4 Resolution in Two Dimensions

3.4.1 Introduction

The resolution of closely spaced pairs of peaks in two dimensions is of similar practical importance to the one dimensional case, applying to the resolution of the images of double stars. It is rather simpler, in that there are fewer variants to be considered: only noise uniform across the image was used, the presentation was always two dimensional with a uniform background, and there was no width parameter. Two forms of image were used, continuous tone and contour, the former being positive (that is, bright stars on a dark field) to give best contrast. About 50 images, each with 9 double stars and each plotted both ways, sufficed for the main part of this work.

The basic method was to generate an image of double stars on a 3 by 3 grid, all at the same ratio of component brightnesses, and separation increasing by an overall factor of 4 in steps of about 20%. A field of noise was also generated, and both were similarly filtered. Mixing them produced images with various noise levels, which were then plotted. The position angles of the double stars were random, the values being recorded. That the corresponding double stars in all the images from one initial star image had the same angle was no disadvantage provided the assessment was begun with the most noisy.

Ws, Wf are used with the same meaning as before. Sn refers to the weaker component. Separations are expressed relative to sigma for the star profile, usually after filtering. Ratio refers to the ratio of the strengths of the component peaks.

The results obtained are summarised in Fig.3.4a. Examples showing the appearances of double stars under different conditions appear in Pl.3.4a and Fig.3.4b.

3.4.2 Continuous Tone Images

In this work the filter mostly used was gaussian with Wf/Ws=½, chosen to give only a slight loss in effective Sn but sacrificing a predicted 11% on resolution. Tests made with Wf/Ws=1 and without filtering confirmed the expected loss of resolution. Even unfiltered (Ws=2) the apparent loss in Sn was slight, and would

for most purposes be negligible, so in practice Wf/Ws=¼ would be more appropriate where resolution was critical, the loss being only 3%. Separations have been expressed relative to the profile sigma after filtering, and thus directly represent the performance relative to Ws when such slight filtering is used.

Tests were also made of square filters, 2.5Ws and 1.5Ws on a side. Their performance was indistinguishable from gaussian filters of similar effectiveness, apart from the appearance of the noise.

Four degrees of resolution were considered. The first was clear resolution, the star images appearing separate but just touching. The second, resolution into components just separate enough for their spacing to be measurable and their ratio estimated. Third, the image just but reliably discernable to be out of round and hence, most simply, double. Finally, the same but with a moderate chance of error.

The second represents the quality of resolution normally required. It was found, in the absence of noise, to need a separation of about 2 at moderate ratios, 1.7 at a ratio of 1 rising to 2.4 at 9; at 27 it has increased to 2.8. It is very similar to the results obtained for spectral lines, and is substantially better than the separation required for zero dip between the peaks, especially for unequal pairs.

The first is essentially complete resolution, and is more than is normally needed. The separation required, in the absence of noise, was greater than in the previous case by about 1, ranging from 2.9 to 3.5. It is very similar to that given by the zero dip criterion at ratios from 3 to 27, and substantially worse at lower ratios. It is seen that the zero dip criterion is reasonable but rather conservative: for markedly unequal doubles it corresponds to complete apparent separation of the components.

Reliable discrimination of the distortion of a star image produced by close doubling was found possible to very much closer spacings: less than for measurable resolution by about 1 for a noise free image. The spacing of 0.75 needed at ratio 1 is only 44% of that needed for resolution, though the relative gain is less at high ratios: 1.9 at 27 is 68% of the spacing for resolution. Even at this slight degree of separation it was nearly always possible for ratios of 3 or more to tell the position of the weaker component. Hence some slight deductions at least can be made as to the separation and ratio of a double from the degrees of elongation and asymmetry visible, even at these very close spacings.

Discrimination became only marginally possible when the separation was reduced by quite a small amount, about 20%. At these closer spacings it became quite impossible to discern the position of the weaker component. Although the threshold for discrimination is apparently rather sharp, it must be expected to vary substantially between observers.

In the presence of noise the separations required are greater. The increase in separation was found, empirically, to be proportional to 1/Sn for Sn from 5 to 32. The increases were small, not much bigger than the errors, so the dependence on ratio remains uncertain. The results agree with an increase independent of ratio (from 1 to 27), at Sn=5 of 0.3 for complete resolution, 0.4 for just measurable resolution, and 0.8 for the threshold of discrimination. The general form of these results seems reliable, although the actual figures might be in error by a substantial factor (but not more than 2).

The effect of noise on complete and just measurable resolution is seen to be small: 5 and 10% at Sn=10, which in most cases will be negligible. Visually, the

image does not appear much less well resolved as the noise increases, just becomes harder to see. The threshold of discrimination is much more affected: to hold the loss to 10% typically needs an Sn of 50. However, even at Sn=12.5 the threshold is still about 0.5 below the limit of measurable resolution in the absence of noise.

3.4.3 Position Angle

The accuracy with which the position angle of a just resolved double star can be found is limited when noise is present. An approximate theoretical value can be found very simply if we assume complete resolution, a high ratio, and a separation of 2. Then the error in angle comes mostly from the error in the transverse position of the weak component. This is 1.4/Sn rms, so the rms error in the position angle is 0.7/Sn radians or 40/Sn°. At high ratios the separation will be nearer 3, reducing the error; at low ratios, the position errors for both stars contribute increasing the error by up to 40%. Further, when resolution is not complete the error will be increased. Even so, 40/Sn° seems a reasonable estimate of the rms error introduced by noise in the position angle for close but resolved double stars. If Sn is based not on the actual strength of the weak component but the margin by which the threshold for discriminating doubleness at this spacing is exceeded, the effect of incomplete resolution should be included, at least partially.

It was found that the position angle could be estimated with an rms error of about 10° at Sn=5, 5° at Sn=12.5, and 3° at Sn=32. In the absence of noise, the error was about 2°. This residual component of error depended on the detailed method of measurement: the method used, setting the image to appear parallel to a reference line along side, could be considerably improved. These results were in very satisfactory agreement with simple theory. For double stars that could be just reliably detected as such, the rms error was about 12°. When the angles were estimated by eye, an rms error of 5° was introduced.

3.4.4 Contours

Contour plots gave results very similar to the continuous tone images. Only the differences will be discussed.

Choice of filter was of more consequence. The standard for comparison is again Wf/Ws=½. With a matched filter, the loss of resolution was offset by a much cleaner and easier to read plot, with most of the tiny wiggles smoothed out of the contours. While with no filter, the irregularity of the contours becomes extreme and lots of isolated noise peaks appear, making the plot very confusing indeed. Thus in practice it seems desirable to use as much filtering as possible; certainly rather more than for continuous tone images. Square filters were unsatisfactory, introducing a needless jaggedness into the contours.

The degrees of resolution can't be given quite the same meaning. Interpreting them as similarly as possible, the separations for complete resolution were essentially the same, but just measurable resolution required a little more, about 0.5 (20%), reflecting the lack of help from the peak in the spatial frequency response of the eye. Noise had even less effect than on continuous tone images.

The threshold of reliable discrimination was much the same, perhaps a little worse at low ratios, and better at high (where the eye somewhat lacks contrast

sensitivity), by about 0.2. The effect of noise is similar, but it is much more difficult to decide if an observed slight irregularity of shape is significant or just noise.

This threshold of discrimination was based on quite rapid scrutiny of plots of average quality. With little noise, considerably better performance is possible. This was investigated with a series of noise free plots on a larger scale, examined very carefully. The expected theoretical relation of separation to ratio was confirmed: results will be given for a ratio of 1 only. Discrimination was marginally possible at a separation of 0.3. However, this limit was derived by estimating position angles irrespective of whether the star was apparently double, and a very slight distortion could be detected in this way even when it was insufficient to be seen as such. The separation had to be increased to 0.5 or 0.6 before the distortion could be reliably seen: it is better than the 0.95 obtained previously for reliable discrimination, but not unreasonably so considering the greater care taken.

This test was extended to check the threshold for determining the position of the faint component. Below about 0.6 at 3, or 1.2 at 27, it was quite undetectable: 0.9 at 3, 1.7 at 27 seemed about the limit for reasonable reliability. This agrees with the result that at the normal threshold of reliable discrimination orientation is visible, at least for a ratio of 3 or more.

Discrimination thresholds in the absence of noise are not well determined. Even for a single observer, they can change considerably with time and circumstances: variation over a factor of nearly 2 has been noted. Exact values are clearly not too significant.

3.4.5 Limiting Visual Sensitivity

The discrimination threshold and the sensitivity of the eye to slight departures from circularity were investigated further by different means. Ellipses or single contours about double stars were plotted, 25 to a field, on a Kingmatic flat bed plotter: an example, plotted on microfilm, is shown in Fig.3.4c. Each field included a number of circles as well as contours of different shapes, randomly placed on the grid and with random position angle. The sizes of the contours were randomly varied by a few percent to mask any apparent change in size with shape. Contours were 30 or 40mm across on the plot, sufficient that best sensitivity to shape occurred when viewed from a little more than normal reading distance: about 0.5m seemed best. Angle was measured by first placing a clear plastic sheet with a scribed line so the line appeared to bisect the contour.

To complement the plots, two shape parameters were calculated for contours about double stars over a wide range of conditions. The more important was the ellipticity , the fraction by which the short diameter was less than the long; the other was a measure of skewness. The ellipticity at small separations agreed with the theoretical figure of $0.5R(S/(1+R))^2$ where S is the separation, R the ratio.

The results indicated that the discrimination threshold, for ratios from 1 to 99 and various contour levels, corresponded to an ellipticity of about 2%, agreeing with that determined from the plots of ellipses. It varied considerably between individual tests: values from 0.8 to 2.8% were recorded. No systematic effects were apparent, except for a tendency to record higher values when the observer was tired

or distracted. Considerably higher ellipticity was required before it was possible to say that a contour was definitely not circular: about 4%.

The error in angle was found to be quite accurately inversely proportional to the ellipticity from 1.5 to 15%. At 5%, the rms error was 5°.

The measure of skewness was shown to correctly indicate the visible degree of asymmetry, the threshold for correctly locating the weaker component corresponding to a particular value. The only exception was that at high ratios the contour was beginning to appear a different shape, with a slight bump on one side of an otherwise circular contour: then neither ellipticity nor the skewness parameter could be expected to be adequately descriptive.

The threshold parameter values were combined with the calculated values to give Fig.3.4d. The contour level has been assumed to be optimum, provided this is physically reasonable: it was not allowed to lie below 10% of the peak intensity of the weaker component. The discrimination threshold predicted from the ellipticity is undoubtedly too high at high ratios, where the departure from circularity is mostly local, and the lower curve, from the skewness parameter, will be a better approximation to this threshold. The data were not good enough to show this.

Also plotted is a curve for reliable discrimination, based on 4% ellipticity. It is seen to be roughly comparable to the results from standard smaller scale contour plots more casually examined, but rather better. The difference is probably no more than might be expected: obtaining the best possible sensitivity takes considerable patience and much trial and error to find the best viewing conditions. These factors, added to the considerable improvement with practice and the wide erratic variations, made the work exasperating in the extreme.

Examining the calculated shape parameters shows that neither is unduly sensitive to the choice of contour level. If it is assumed that a contour, to be useful and not unduly affected by noise, will lie between the peak intensity of the weaker component and 10% of that, the choice will not affect either threshold by much more than 10% for ratios up to 100. At low ratios, up to 5, the thresholds will usually lie within about 10% of the best possible. The difference increases, to about 30% at ratio 100, but these improvements occur at very low contour levels (0.01% of the weaker component or less) that are unusable because of noise. Thus in practice choice of contour level is not critical.

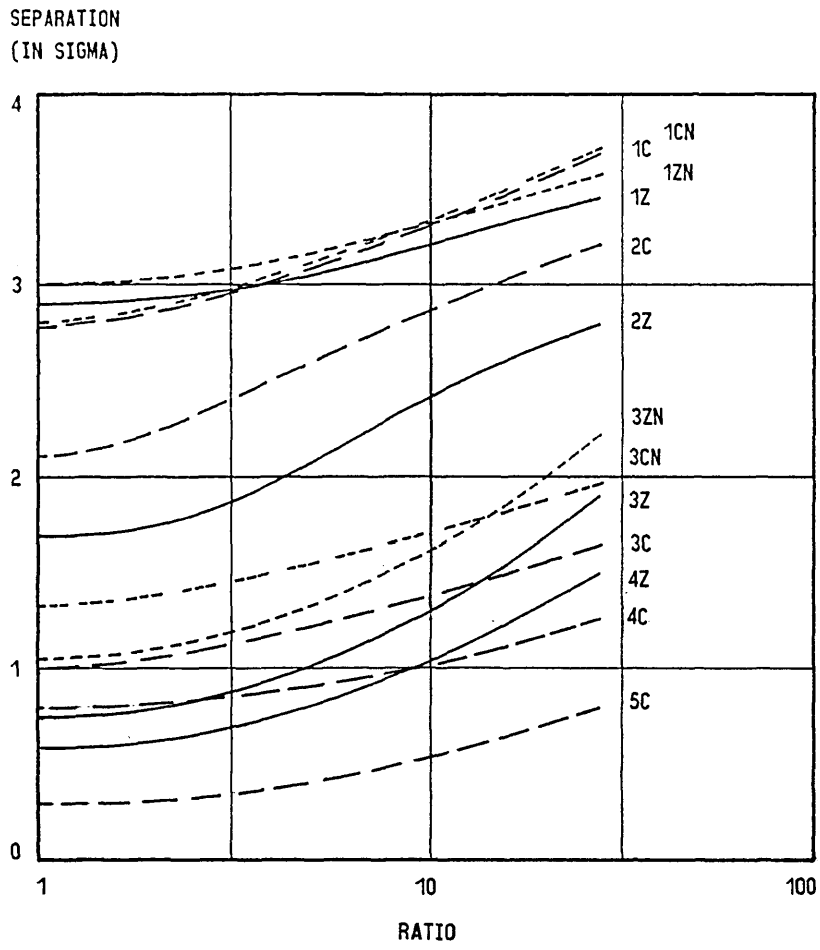
3.4.6 Summary

It has been shown that double stars are resolved into components sufficiently separated to be measurable at a separation of about 2 for ratios up to 3, increasing to 3 at 27. This is comparable to but rather less than the separation required by the zero dip resolution criterion, which seems somewhat conservative. Noise has little effect on the separation required for resolution.

Doubleness is apparent from the distorted shape of the image to much smaller separations, about half those needed for resolution. Under some circumstances the limit can be less by another factor of 2, but this is erratic and the reliability of discrimination is uncertain. At the normal limit it is usually clear to which side is the weaker component, if the ratio is 3 or more. The separation required to discern doubleness can be considerably increased by noise, but if the weaker component is

at least 2 magnitudes above the detection threshold the increase, about 0.2 or less, will usually be negligible.

Continuous tone images and contour plots are comparably effective, resolution being slightly better on the first. Sensitivity to distortion of shape is slightly better on contours, but judging if this is significant or is caused by noise is harder. The filtering required differs. Continuous tone images need very little, $Wf/Ws = \frac{1}{4}$ or the equivalent being sufficient and causing negligible loss of resolution, and square filters are acceptable. Contour plots need more filtering to make them clearly readable, forcing a compromise between clarity of plot and resolution: $Wf/Ws = \frac{1}{2}$ is usually reasonable. Square filters give jagged contours and are unsatisfactory. Choice of contour levels is not critical.



1: COMPLETE RESOLUTION
2: MEASURABLE RESOLUTION
3: RELIABLE DISCRIMINATION
4: MARGINAL DISCRIMINATION
5: EXTREME LIMIT OF DISCRIMINATION

Z: CONTINUOUS TONE IMAGE
C: CONTOUR MAP

N: WITH NOISE, WEAKER COMPONENT Sn 12.5

Fig.3.4a Resolution in two dimensions: double stars.
Accuracy: errors as large as 0.3 in separation are possible in some cases, but relative errors are much less, usually under 0.1.

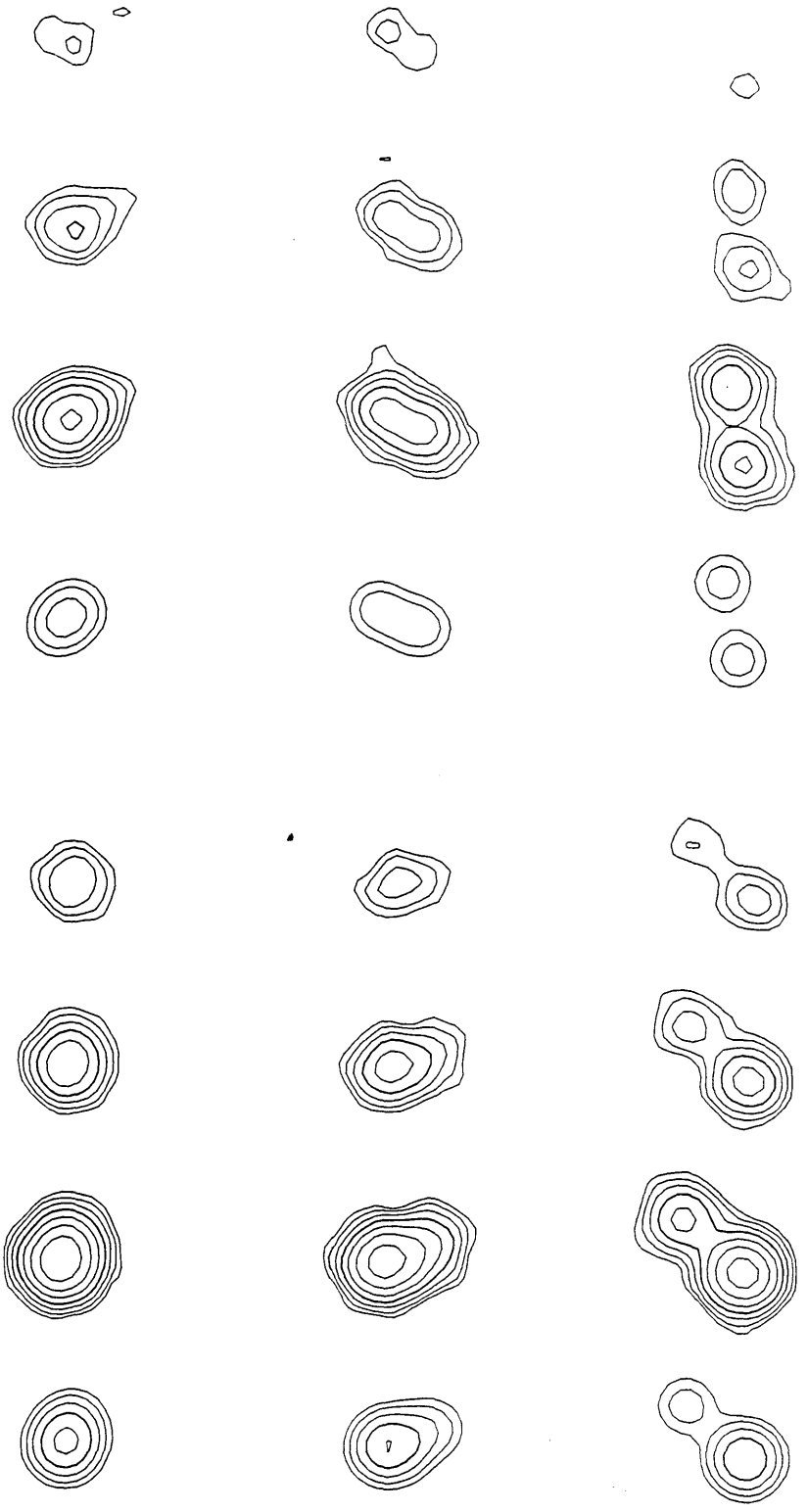


Fig.3.4b(i) Contour plots of double stars: appearance with noise.
Separation relative to profile sigma after filtering (l to r): 1.3, 2.1, 3.6.
Matched filter Sn of weaker component (t to b): 5, 12.5, 31, very high.
Upper: ratio 1. Lower: ratio 3.
Filter: gaussian Wf/Ws=½. Contours at ½ magnitude.

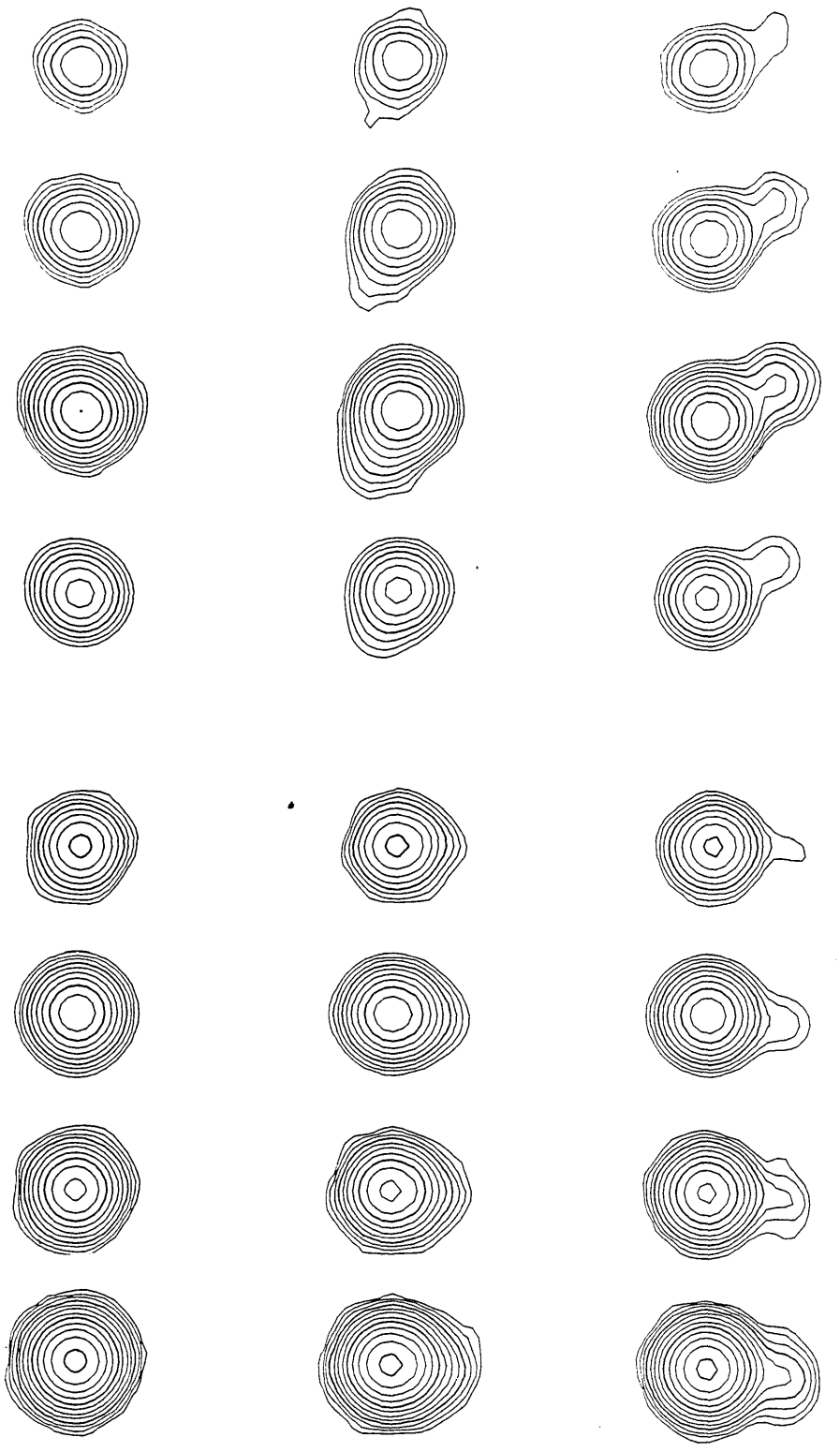


Fig.3.4b(ii) Contour plots of double stars: appearance with noise (cont).
Upper: ratio 9. Lower: ratio 27.

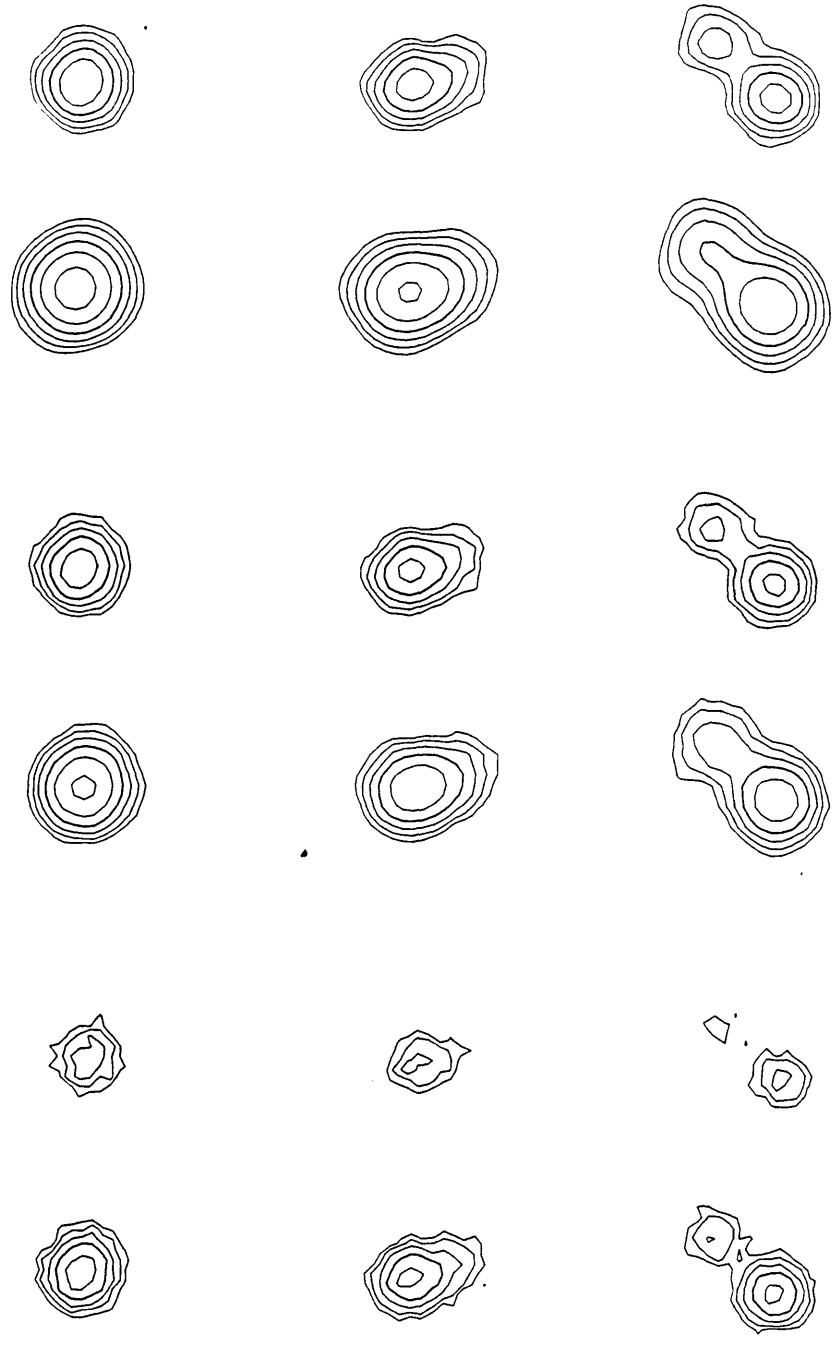


Fig.3.4b(iii) Contour plots of double stars: effect of filter.
Separation relative to original profile sigma (l to r): 1.4, 2.4, 4.0.
Filter (t to b): gaussian Wf/Ws=½, 1; square, side 1.5, 2.5 Ws; none (Ws=2 pixels); none (at 1 magnitude brighter).
Ratio: 3. Matched filter Sn of weaker component: 12.5.

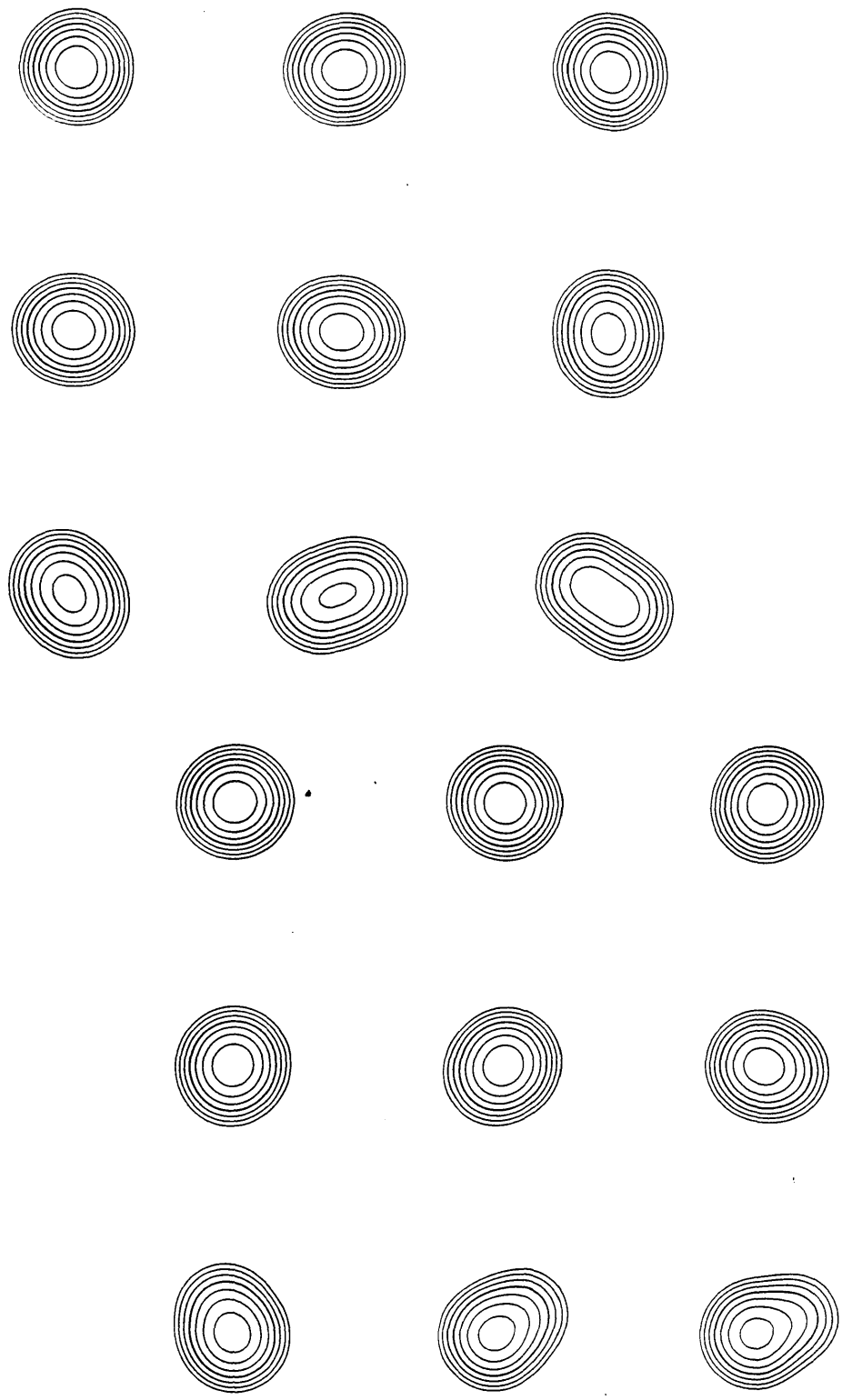


Fig.3.4b(iv) Contour plots of double stars: appearance at small separations in the absence of noise.
Separation relative to profile sigma: 0.5, 0.6, 0.7, 0.85, 1.0, 1.2, 1.4, 1.7, 2.0.
Upper: ratio 1. Lower: ratio 3.

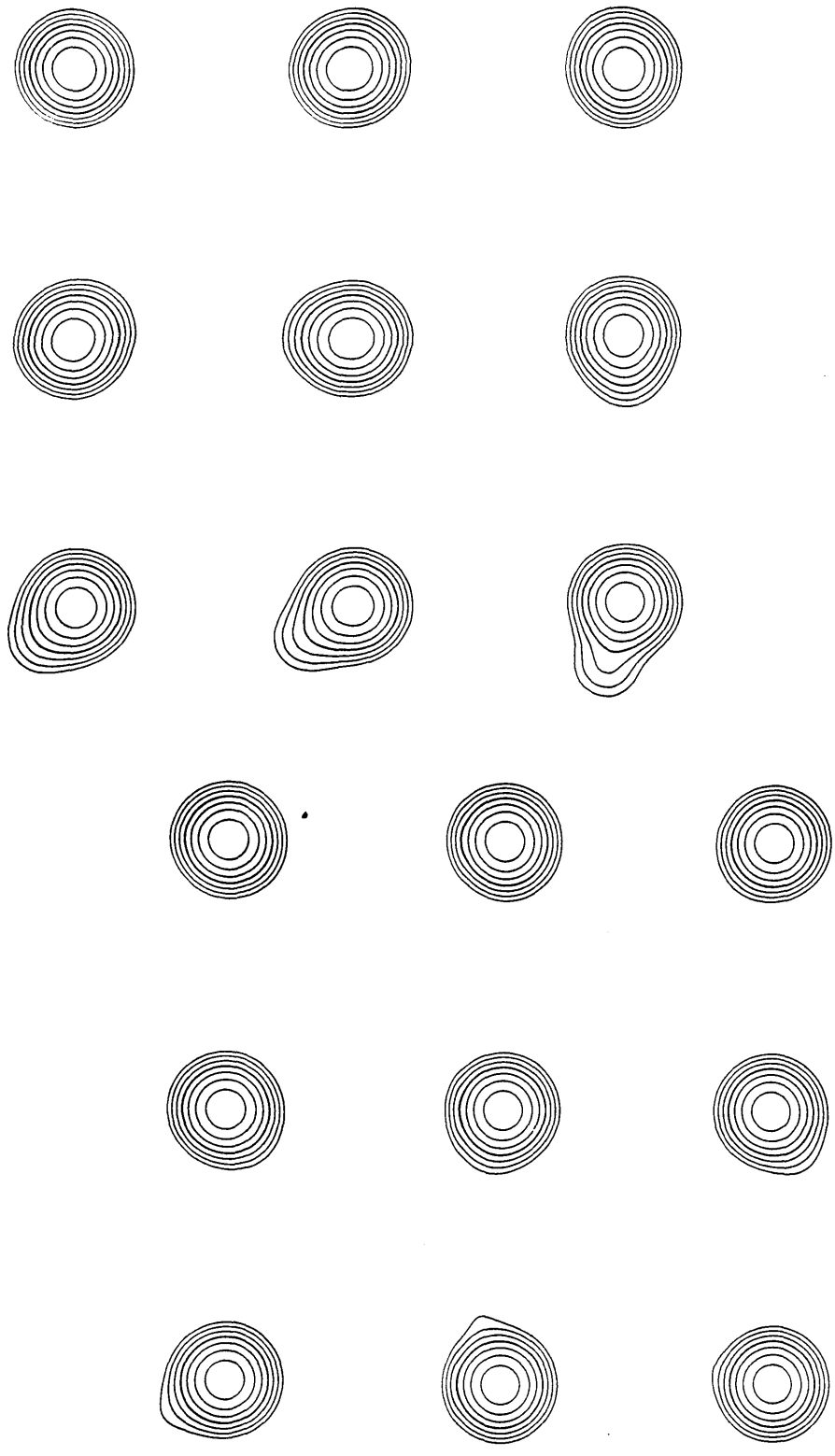


Fig.3.4b(v) Contour plots of double stars: appearance at small separations in the absence of noise (cont).
Upper: ratio 9, separation 0.6, 0.7, 0.85, 1.0, 1.2, 1.4, 1.7, 2.0, 2.4.
Lower: ratio 27, separation 0.7, 0.85, 1.0, 1.2, 1.4, 1.7, 2.0, 2.4, 2.8.

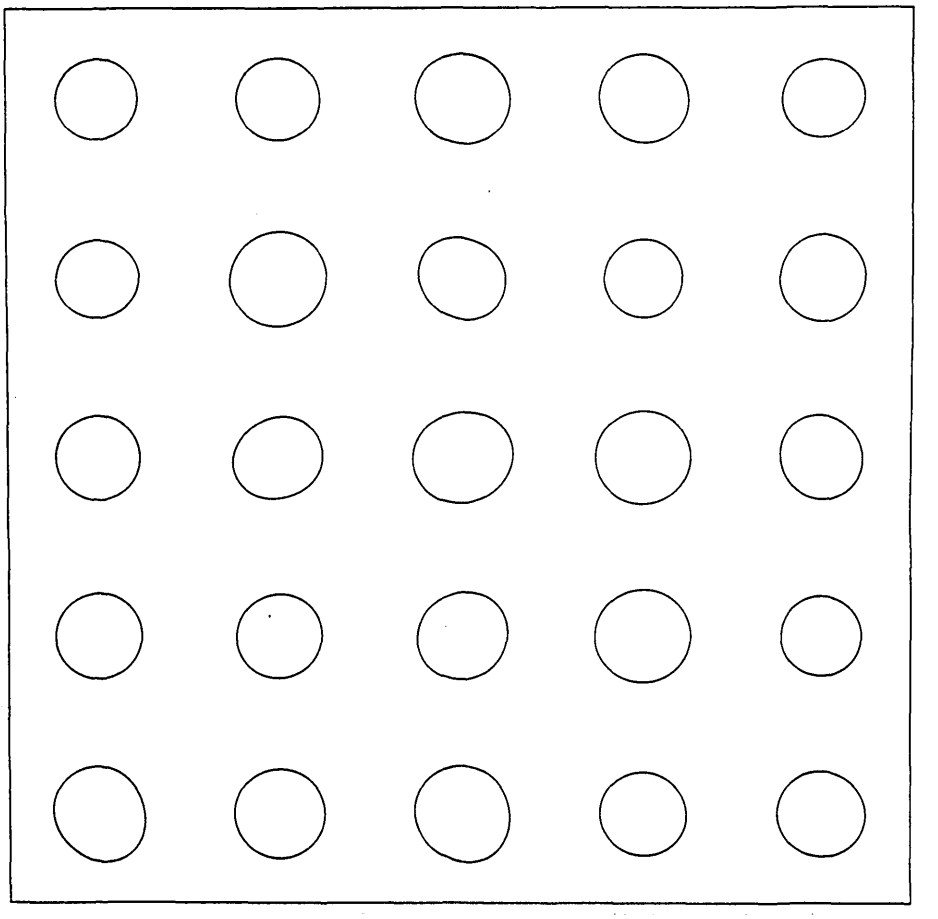


Fig.3.4c Sample frame for testing sensitivity to non-circular contour.

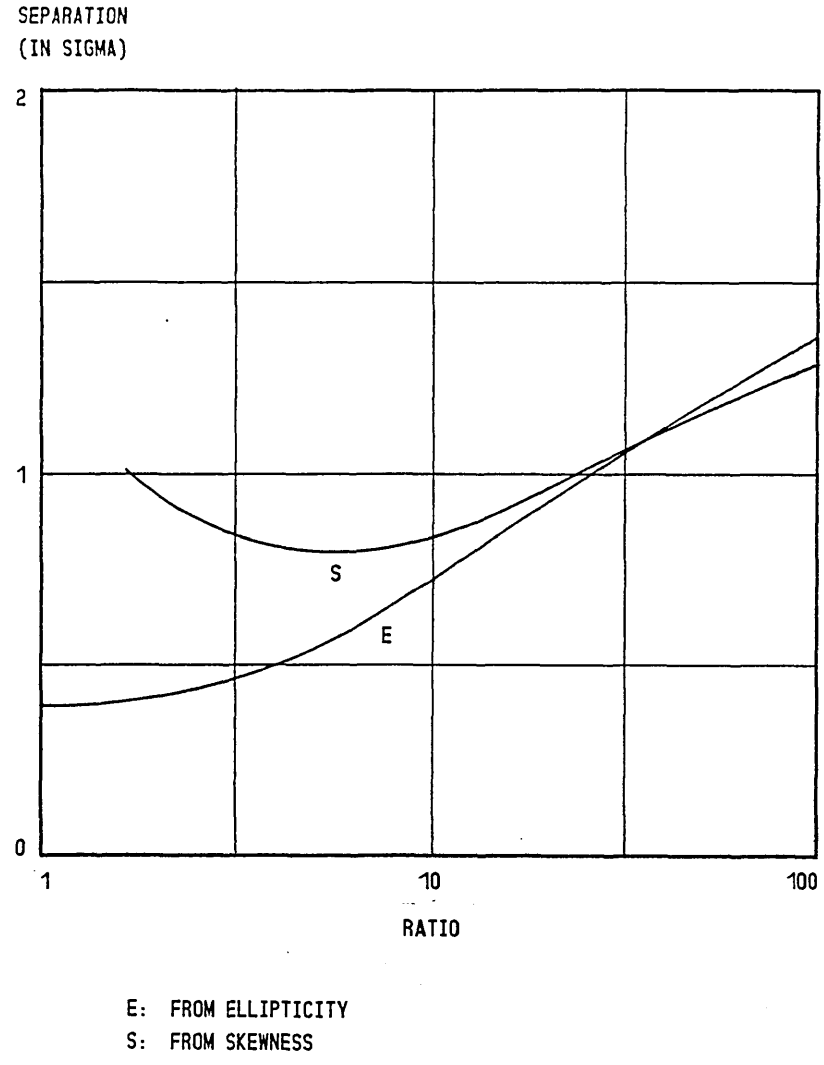


Fig.3.4d Visual sensitivity to distortion of contour of double stars.
Derived from ellipticity (2%) and skewness thresholds.

3.5 Deconvolution in One Dimension

3.5.1 Background

The standard method of improving the sharpness of a spectrum or similar one dimensional image is by partial deconvolution, usually just called deconvolution. One way of implementing it, its properties, and its possible uses will be discussed.

The observed image is assumed to be formed by the convolution of a sharp image with a point spread function. Usually this describes an instrument's response to a sharp input (that is, to a Dirac delta function), but it may be given other meanings, such as the spread in radial velocity of the stars in a globular cluster which doppler broadens the spectral lines in the integrated light. This assumption includes an assumption of linearity, and caution must be used if deviations from it are significant, such as density dependence of the spread function for a photographic emulsion.

This convolution is equivalent to a multiplication in a fourier transform (frequency) space. It is apparent that another multiplication by an inverse filter could cancel the effect of the first, and restore the original image. This would be complete deconvolution, and is not usually possible. If any spatial frequency has been reduced to zero amplitude, it obviously cannot be restored to its original level. Even if the point spread function does not have a zero in its frequency response, such as would be introduced in a spectrograph by an evenly illuminated slit, the response will still usually tend increasingly rapidly towards zero with increasing spatial frequency, and the inverse response would have to rise increasingly rapidly towards infinity and its exact implementation will be impossible.

Partial deconvolution usually will be possible, in which the frequency response corresponding to the point spread function is only partly cancelled, and the combination appears as a point spread function narrower than the original. It will not, except in abnormal cases, be possible to produce a spread function of exactly the same shape as the original. The combined frequency response will have to cut off more abruptly than the original towards high frequencies, which tends to introduce ringing or overshoot into the spread function. Thus to preserve a gaussian shape to a spread function would call for an exponentially rising frequency response; reducing this to zero at high frequencies introduces overshoot; and the lower the cutoff frequency the greater the overshoot, the spread function tending to a sinc profile.

The only practical case in which the method is not possible is when the frequency response falls rapidly to a zero at a relatively low frequency. Since the zero cannot be removed, there is little scope for improving the frequency response. This would arise when the point spread function is close to a rectangular profile, as in spectrography with a wide slit. When the profile is appreciably broadened by other factors (such as the optical properties of the spectrograph and the limited resolution of the photographic plate) the frequency response drops earlier relative to the zero, and deconvolution can effect a significant improvement.

Partial deconvolution in this fashion is the application of a linear spatial filter with a frequency response tailored to convert the point spread function convolved with an ideal image into one apparently narrower. It is distinct from the other use of spatial filters to emphasise high frequencies relative to low, in which supression of low frequencies leaves fine details visible with increased contrast but otherwise unaltered. It is unlike some other methods of deconvolution, and the more modern maximum entropy methods that can be used to improve resolution, in its linearity, which enables the processed image to be interpreted in the same way and with the same reliability as the original.

3.5.2 Method

There are two ways of implementing a spatial filter: convolving the image with the filter point spread function (or impulse response); or taking the fourier transform of the image, multiplying by the filter frequency response, and transforming back. The direct, convolutional, method was chosen as simpler to implement. Any difference in efficiency is of slight consequence, as in one dimension the computation in either case is trivial.

The assumption was made that the point spread functions were symmetric, which allowed cosine transforms to be used and phase to be ignored (being zero). This is reasonable for most truly spatial data, though instrumental aberrations could introduce asymmetry, and so could the temporal response when spatial data is scanned. The modifications to handle the general case are straightforward, and for slight asymmetry the properties will be little altered.

The process will be described in terms of a given input line profile and the corresponding profile when deconvolved. For a delta function input before broadening these are the same as the point spread functions, and make for a more concise description, and, interpreted as a sharp line in a spectrum, a more meaningful one.

The procedure was to first select a frequency response corresponding to a suitable deconvolved line width and shape, especially overshoot. The fourier transform of the given line profile was then taken, and divided into the desired frequency response to give the response of the necessary filter. The fourier transform of this gave the filter to be convolved with the image. This filter, being unnecessarily long, was reduced in length by multiplying by a windowing function.

The input line profile used was gaussian. Except as indicated in the previous section this is not critical. In practice the actual shape from theory or observation could be substituted without much affecting the properties of the method, provided the shape was not radically different.

The selected frequency response was varied in shape to control the overshoot of the deconvolved line. The extremes were rectangular, giving 21.7% overshoot, and cosine squared, giving 2.7%. The exact shape was not found to be important, provided it was intuitively reasonable; the properties of the filter were closely correlated with the overshoot produced, and little affected by the shape which produced it. In all cases the chosen response was identically zero above a cutoff frequency, so the filter would have a corresponding cutoff and be certain not to rise indefinitely at high frequencies.

The cutoff in the frequency response of the filter implies an impulse response extending to infinity. If the length of the filter convolved with the image is less than twice the length of the image, this arbitrary truncation causes spurious responses to an input line, separated by the half-length of the filter. If not, the regions of ripples caused by the abrupt truncation of the input image at each end extend throughout the deconvolved image. These effects are avoided by multiplying the filter by a windowing function which goes smoothly to zero outside some distance from the centre.

The exact form of the windowing function is not critical. If it is considerably wider than the minimum necessary it is enough that the function and its slope both vary smoothly and not too rapidly, and that if it is truncated at all it has already fallen to a very low value. In this case it can be flat topped, and will cause negligible departure from the desired response.

If the length of the filter is to be reduced to a minimum, the windowing function needs a little more care. The shape used here was a cosine curve convolved with a narrower gaussian to give a profile rather like a gaussian but fatter at the top and with reduced wings, which were truncated at a low value. Similar shapes gave much the same results, but a little tweaking to get the narrowest possible filter seems possible. A windowing function of this sort modifies the deconvolved line profile by increasing the overshoot, and this should be allowed for in choosing the initial frequency response.

Two families of deconvolution filters are shown in Fig.3.5a. Sharpening is expressed by the improvement in resolution for double line profiles with equal components, as measured by the zero dip criterion, and is in the range 1.2 to 2. Note that the lower overshoot requires response to higher frequencies, and a greater gain at high frequencies relative to low. For sharpening by a factor of 2, the peak high frequency emphasis is by a factor of about 50,000, while for 1.4 it is only about 40.

An example of deconvolution applied to a double line in the presence of noise is shown in Fig.3.5b, with corresponding continuous tone images in Pl.3.5a. It is clear that a double line with no indications of doubleness beyond its greater than normal width can be cleanly resolved by quite moderate sharpening (by 1.4), and even a sharpening factor of 1.2 gives a considerable improvement. The main limitation of the method is quite clear: the drastic increase in noise when much sharpening is attempted, so that at a factor of 1.7 the lines are completely lost in the noise, despite an original signal to noise ratio for each component of about 280 with a matched filter.

3.5.3 Noise

It is apparent that if the filter has unity response at zero spatial frequency the mean signal level is unaltered, and the peak amplitude of a line will only be increased by approximately the sharpening factor. On the other hand the noise, if we assume it to be uncorrelated between sample points, is spread uniformly over the frequency spectrum, and the high gain of the filter at high frequencies leads to a high noise level in the filtered image. The greater the sharpening, the greater the high frequency gain, and the greater the noise level. This increase is very rapid, and ensures a rapidly deteriorating signal to noise ratio at high sharpening factors. The deterioration is worse when only a slight overshoot is permitted: to reduce the overshoot while preserving the narrowness of the line requires the restoration to high level of higher frequencies, and hence increases the noise.

To quantify this deterioration a measure of the degree of sharpening is needed. The ratio by which the resolution of equal components, indicated by the zero dip criterion, is improved is the measure chosen. This is a realistic and practically useful measure, but may be slightly pessimistic. When a pair of unequal components is considered, the improvement in resolution is slightly better than by the sharpening factor defined for an equal pair, because the deconvolved line profile falls more sharply in the wings than the original gaussian line. For a 3:1 ratio pair and 4% overshoot the sharpening is 16% better than for an equal pair. It is considered that the sharpening factor defined for equal components is adequately representative if it is remembered that, taken over pairs of various ratios, the effective sharpening factor is typically some 10% greater.

There is also a need to define the amount of overshoot that is permissible. It is found, roughly, that a double line can be clearly resolved without undue interference from the ripples in the wings if the overshoot is no more than about half the strength of the weaker component relative to the total. Thus, for equal components, an overshoot of 25% is permissible, and so this is the maximum allowable for any use. At 3:1, 12% is acceptable, and this can be regarded as the greatest for any normal use. At 9:1 the limit is 5%, and this should be adequate for general purposes. The range of interest is thus 5 to 25%, which corresponds to a corrected frequency response from cosine squared to rectangular, allowing for a small increase in overshoot produced by windowing. The case of very low overshoots, needed for resolving very unequal pairs, and the associated problems was not considered.

The deterioration produced in signal to noise ratio, compared to a matched filter, is shown in Fig.3.5c against sharpening factor, for different degrees of overshoot. This was determined by constructing and numerically evaluating several series of filters, and the method checked by applying sample filters to line profiles and random noise. The rapid and increasing rate of deterioration in signal to noise with increasing sharpening factor is clearly evident. There was no difficulty in producing filters capable of sharpening by factors at least as high as 2.4, and that worked perfectly, but with signal to noise degraded by a factor of 30,000 or more this was clearly going beyond the bounds of practical utility.

There is no sharp limit to the value of signal to noise degradation factor that may be tolerable, in that it will depend on the details of the experimental system. Some examples will suggest values that might be usable.

Most simply, consider a spectrum recorded on a photographic plate. For an absorption line recorded to a depth of 50% in a background of D=1, with the line 10mm long and a width of 1mm we can get a signal to noise of no better than about 2000 even on a very fine grain plate such as IIIa-J. Even on a holographic plate such as 649-F it will be less than 8000 (Kodak, 1971). Since we need a final S/N of about 10 to be useful, it is evident that the degradation factor must be less than 1000, and for realistic astronomical uses no more than about 100.

Next, consider scanning across a 50% deep absorption line with a single channel photon counting detector. For 100 samples, each taking 10sec, with a count rate of 1 million per second on the background, we can expect a S/N of about 8000. This would allow a degradation factor about 1000, but represents a performance not attainable on any object except the sun.

Scanning has a second source of noise, that applies even when a plate is being scanned. If the scanning mechanism has a positioning uncertainty of 0.5 microns rms, in a one dimensional scan across a line 1mm wide, sampling every 10 microns, noise is introduced that would limit the S/N to about 25,000. If the positioning error is correlated between samples, this might drop to about 10,000, and there would be a similar limit on a two dimensional scan. This again suggests a degradation factor limited to about 1000 in a near ideal case, and more usually to about 100 (the noise gets much worse for a narrower line).

For a practical plate scanner, working at a moderate speed of 100 samples per second, digitising density to a 3 digit accuracy, and scanning a spectrum with 100 lines in a time of 20mins, for a line depth of about half maximum density there is an instrumental limit of a S/N of about 7000. While this is in no sense a hard limit, it is representative of what might be expected in routine scanning of spectra. Again, it would limit the degradation factor to something less than 1000.

These examples indicate that a signal to noise degradation factor as large as 1000 might be usable in very special cases, but that it would be unusual for more than 100 to be usable in any normal astronomical application. Sharpening factors could thus in the extreme be 2.1 with 25% overshoot, or 1.85 for 5%, but for astronomical use 1.8 with 25% overshoot or 1.65 with 5% would be more representative limits.

It should also be noted from the curves that a slight improvement in resolution is obtained at rather small cost: a sharpening factor of 1.2 with 5% overshoot degrades signal to noise by about 3, and of 1.35 by about 8.

3.5.4 Other Properties

The length of the filter is reduced by the windowing operation, and the extent to which this is possible depends on the sharpening and overshoot required. The mimimum length depends on what is considered acceptable deterioration in the shape of the deconvolved profile, and the care with which the design of the filter is tweaked. With no special adjustment it was found that there was a just noticable deterioration for 5% overshoot filters at half-lengths of 3½, 4, 5½, 8 times the input line profile sigma for sharpening factors 1.2, 1.4, 1.7, 2.0. At about 20% shorter the deterioration was the maximum tolerable. For 25% overshoot filters the half-length could be very slightly, about ½ sigma, less. The effect of reducing the length of a filter is shown in Fig.3.5d. This also illustrates how simple truncation is unacceptable even when the length is considerably greater.

A question of considerable practical importance is the sensitivity to a difference between the assumed input line width and the actual. The effect of such a mismatch is shown in Fig.3.5e. It is found that a small error produces a change in the sharpening and overshoot without affecting the general character of the filter. Indeed, a deliberate mismatch is a satisfactory way of slightly adjusting the overshoot produced by a filter. The effect of a mismatch increases with sharpening factor: a 3% error is only marginally acceptable at 2.0, and 10% at 1.4 (for low overshoot filters). For practical sharpening factors under 2, an error in the assumed width of 5% or more is tolerable. It should easily be kept to less than this if the line width is instrumental and can be determined directly, and should tolerate any variations across the image and between observations. In the worst case of completely unknown width, it would need only a few trials to find the best value, provided a correctly deconvolved image can be recognised (possible, for example, if a spectrum contains an isolated sharp line).

A related practical problem is the sensitivity to an error in the shape of the input profile. This was analysed by assuming a quadratic distortion term such as might be introduced by a non-linear stage in the image recording process (such as a photographic plate). Characterising it by the maximum departure from the mean slope of the transfer characteristic over the range used, a 10% deviation was found tolerable at a sharpening factor of 1.4, 5% at 1.7, and 1% at 2.0 for low overshoot filters. This is based on the appearance of a deconvolved line, and is illustrated in Fig.3.5f. When the resolution of a double line is considered, distorted in the recording process, considerably more distortion is tolerable: at 2.0, 5%. This suggests that at low sharpenings, up to 1.5, slight deviations from the assumed line shape, and non-linearities in the recording process, are no problem. At greater sharpening, up to the practical limit of something under 2, care is needed and any photographic non-linearities may need compensating, but the sensitivity is not such as to pose difficulties.

The use of sampled data poses the usual problems. If the cutoff frequency of the deconvolution filter exceeds the Nyquist frequency, aliasing will occur, and unless very slight will completely destroy the image. This is seen in Fig.3.5g. As a rough guide, results are satisfactory if the initial line sigma in sampling intervals is at least 0.9 times the sharpening factor. There is also a slight variation in the apparent peak value of the deconvolved line with its position relative to the sample points. This is up to 2% from the mean for an initial line sigma in sampling intervals of 2 times the sharpening factor, and 10% for 1. This is purely an effect of sampling, as can be shown if the deconvolved lines are filtered back to a greater width, when the variation disappears. This is illustrated in Fig.3.5h.

3.5.5 Utility

It has been shown that partial deconvolution of one dimensional data is a simple procedure free from any difficulties other than the deterioration in signal to noise, and capable of a worthwhile gain in resolution. The traditional rule of thumb that sharpening by a factor of two is possible is shown to be slightly optimistic in that it requires original data of extremely high signal to noise, but a factor of 1.5 to 1.8, corresponding to a S/N degradation of 20 to 100, is compatible with high quality data obtained by normal astronomical techniques. The sharpening factor would be a little better if based on the resolution of unequal line pairs, but still no

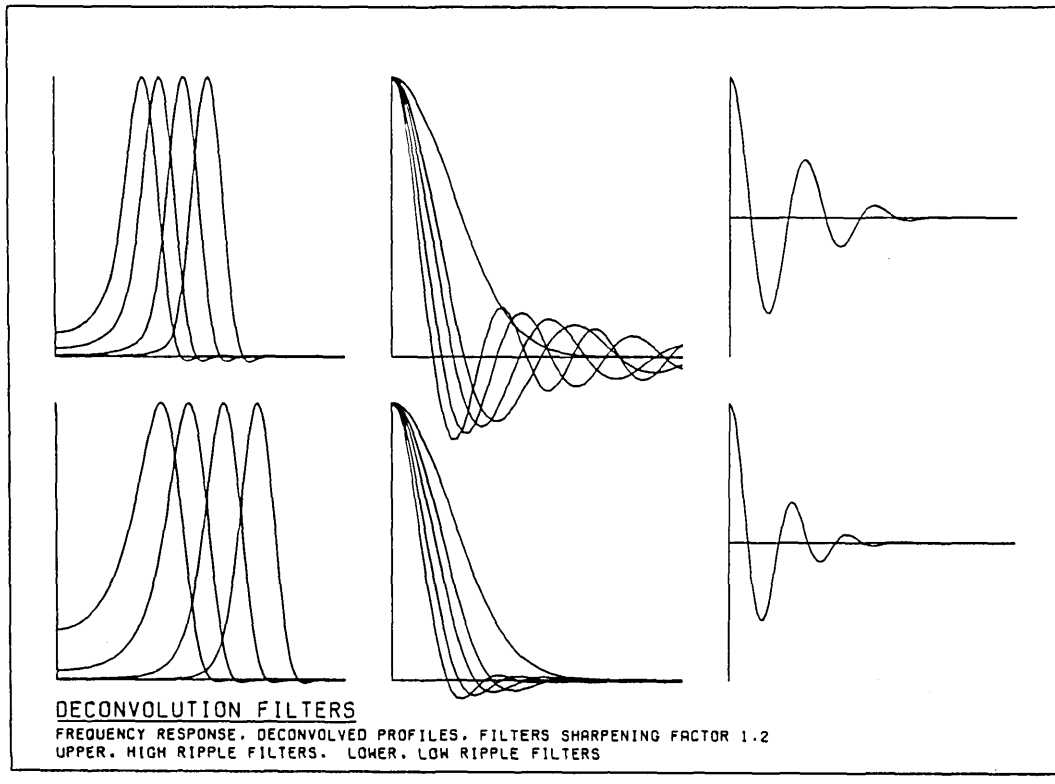
more than 2. Despite this potential the technique is almost unused, in optical astronomy at least.

The reasons are easily seen when it is considered that most one dimensional data in optical astronomy are spectra obtained with spectrographs in which resolution can easily be gained at the expense of sensitivity by narrowing the slit, changing to a finer grained plate, or changing the grating. Consider as an extreme case doubling the resolution by doubling the dispersion with a different grating. To cover the same spectral range will now take two exposures, so the available observing time will have to be split between them. The image will be only half as bright because of the higher dispersion, and together these factors will double the noise level relative to a background continuum. A narrow line, substantially finer than the instrumental resolution, will appear at nearly twice its previous depth, so its S/N is virtually unchanged. And even for a broad line the deterioration in S/N is only a factor of 1.4. Thus the increased resolution is achieved virtually without loss. Even if the grating can't be changed, the slit can be narrowed for a slight increase in resolution. At a typical setting the slit width contributes equally to the total of the other contributions to the instrumental line width. If the slit width is halved, the resolution can be increased 25% for a degradation in S/N of 1.4, or less for a sharp line.

These arguments are very approximate, and spectrography is subject to practical difficulties and limitations, but they are enough to show that it is the usual case that any slight gain in resolution obtainable by deconvolution is obtainable instrumentally at much lower cost in noise. There remain a number of cases in which deconvolution may be the only available method. One is when the available instrument is already being pushed to its maximum resolution. A scanning, high resolution Fabry-Perot interferometer is a common example, the resolution usually being fixed by the construction. The analysis of existing data, such as spectra of novae, is another case: a slight sharpening would often be possible.

There are also all those cases in which the broadening to be removed from the spectra is produced in the source, and deconvolution (or at least some method of processing the observed data) is the only option. These need particular care in that there is usually no easy way to determine the spread function to be deconvolved; the content of high spatial frequencies in the spectrum may be a guide, or the width of an isolated line, or knowledge of the expected deconvolved appearance. They include such diverse possible uses as resolving slight Zeeman splitting of stellar lines, revealing more kinematic fine structure in lines from emisssion nebulae, and giving improved sensitivity for for detecting double-lined spectroscopic binary stars.

Within this rather limited range of uses the technique seems valuable. Its lack of use must be due in part to the scarcity of fast plate scanners linked to convenient interactive image processing systems. It may also be that many astronomers are not familiar with a method that would only occasionally be useful.



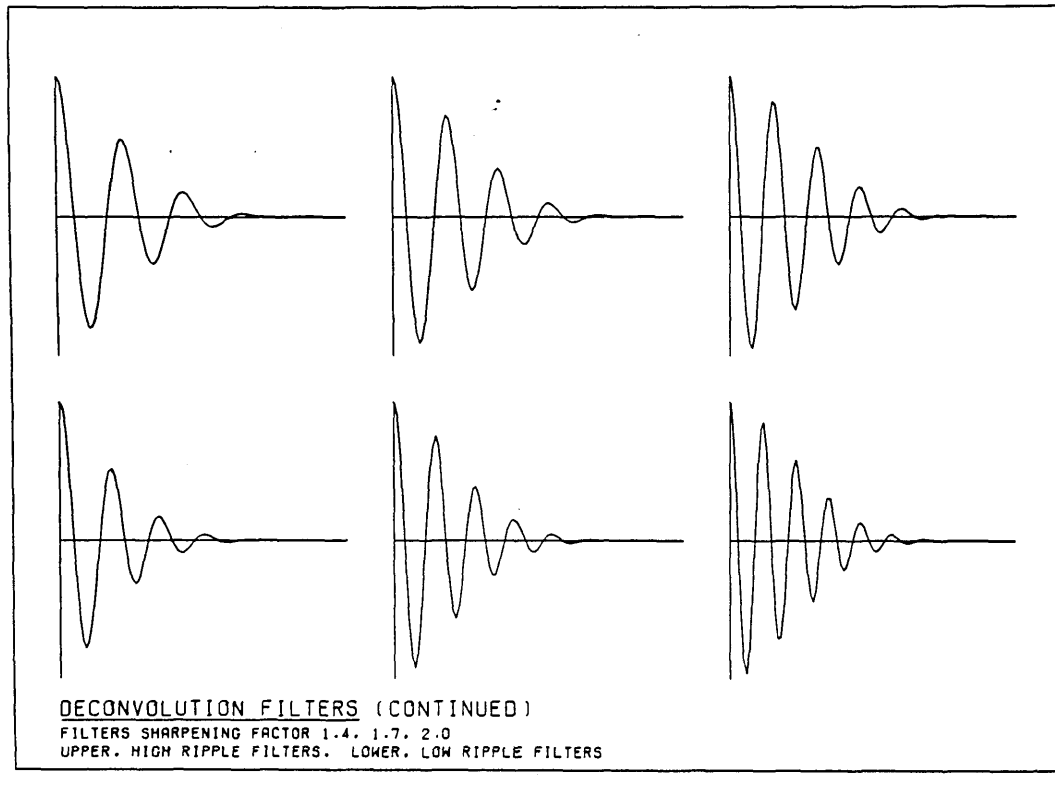


Fig.3.5a One dimensional deconvolution filters.

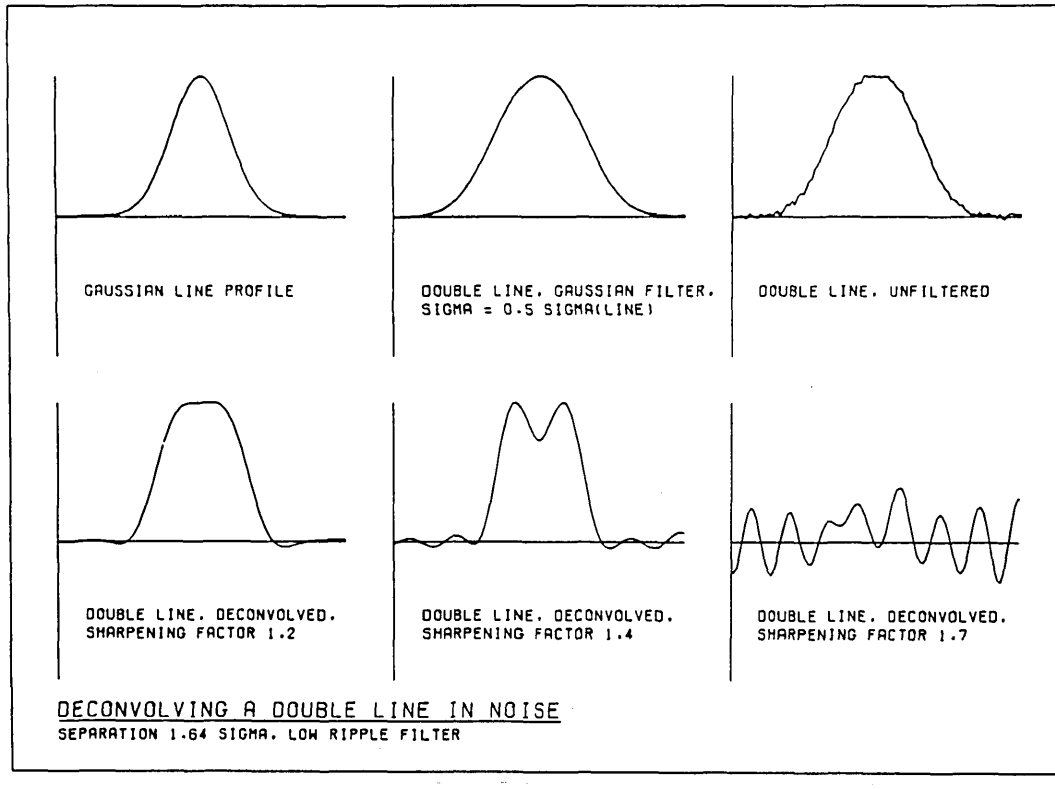


Fig.3.5b Example of deconvolution in one dimension.

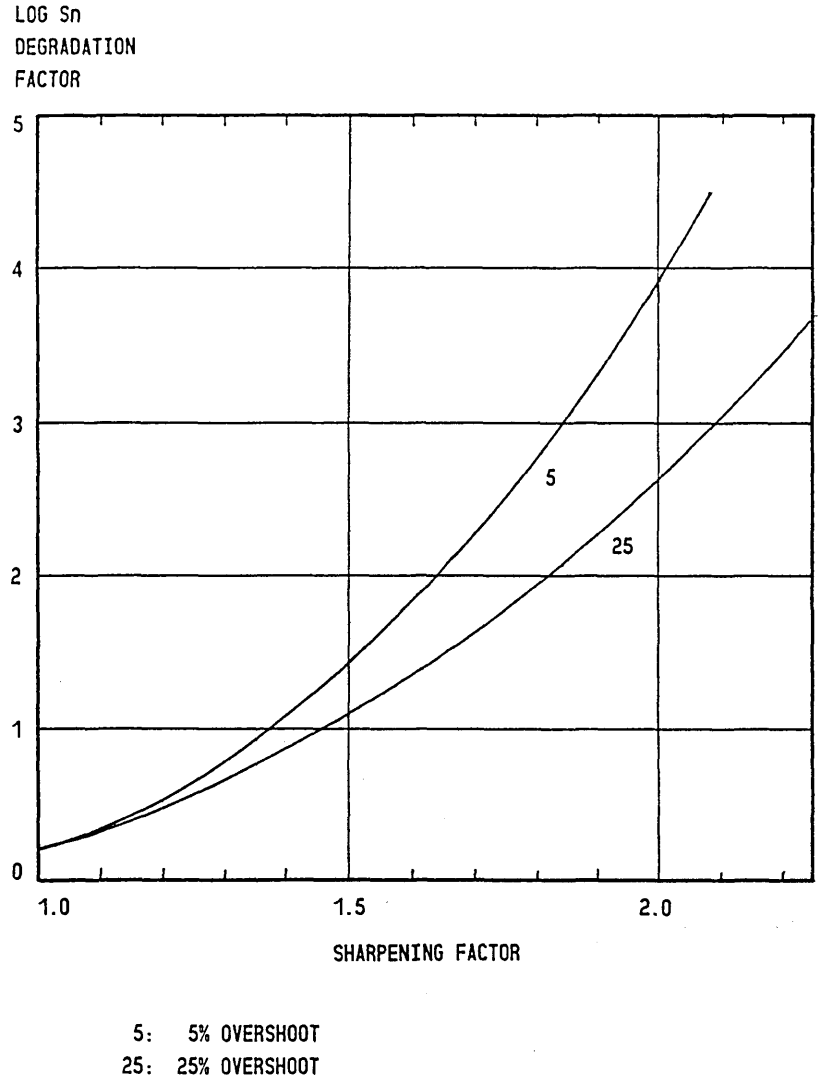


Fig.3.5c Signal to noise degradation in one dimensional deconvolution.
Accuracy: about 1% in sharpening factor plus 10% in degradation factor (2 sigma).

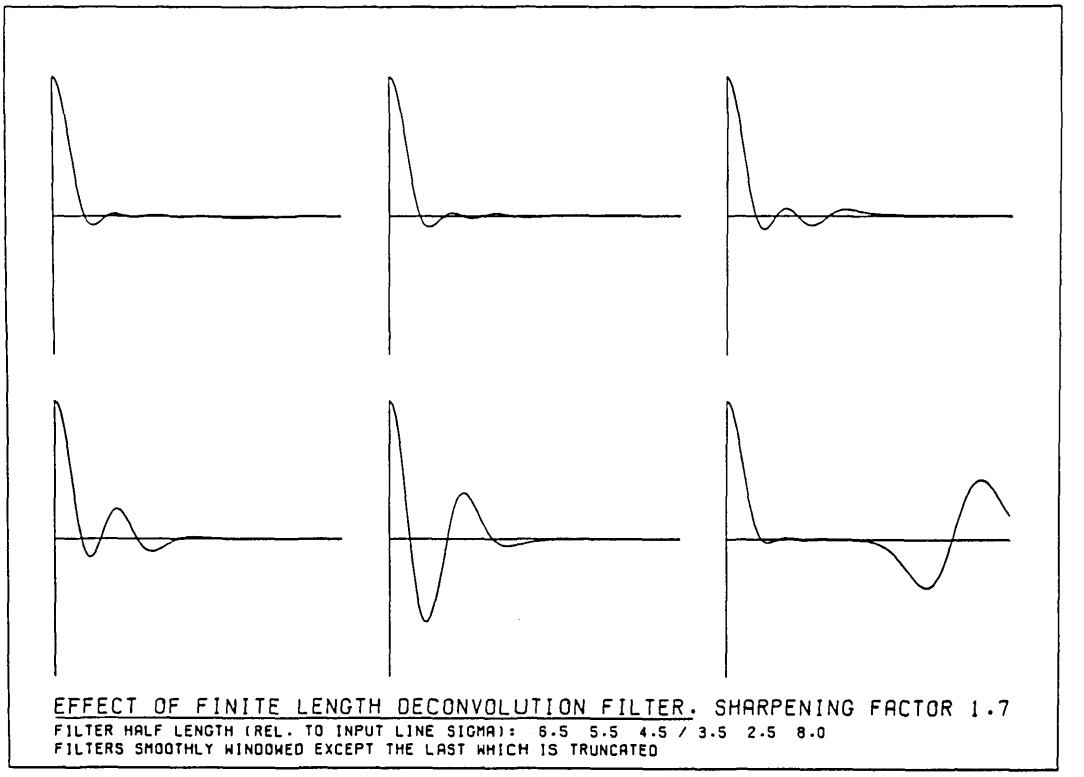


Fig.3.5d Effect of filter length on one dimensional deconvolution.

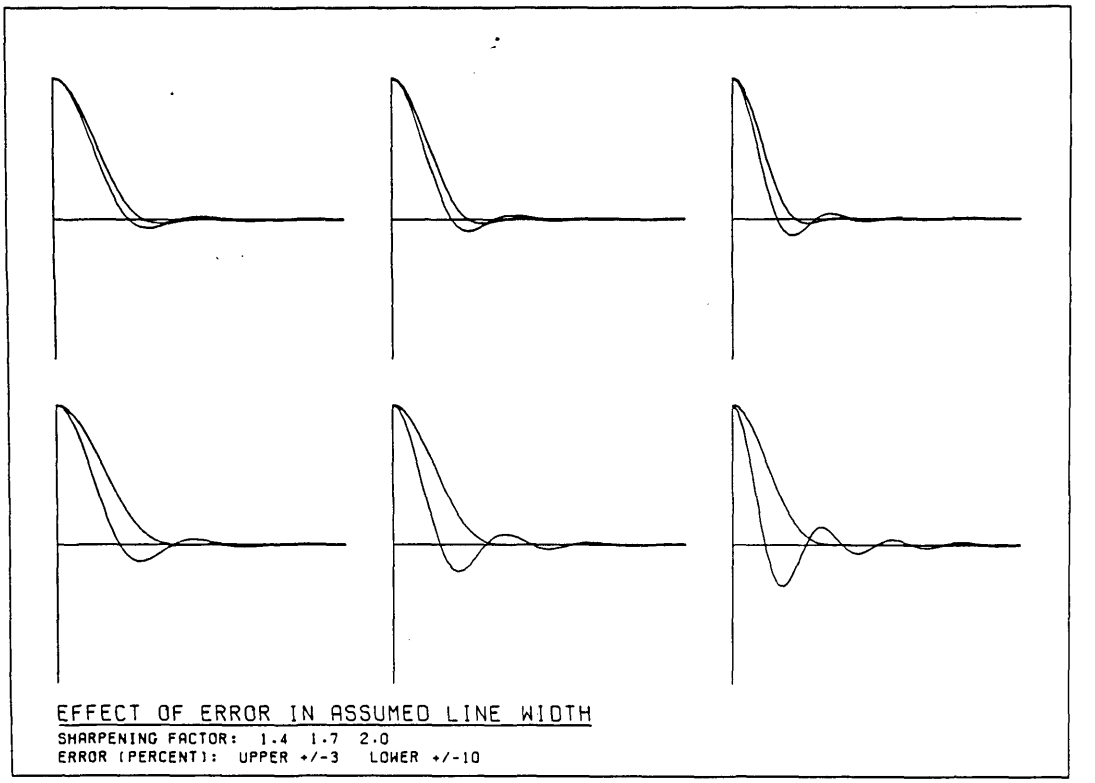


Fig.3.5e Effect of error in assumed width on one dimensional deconvolution.

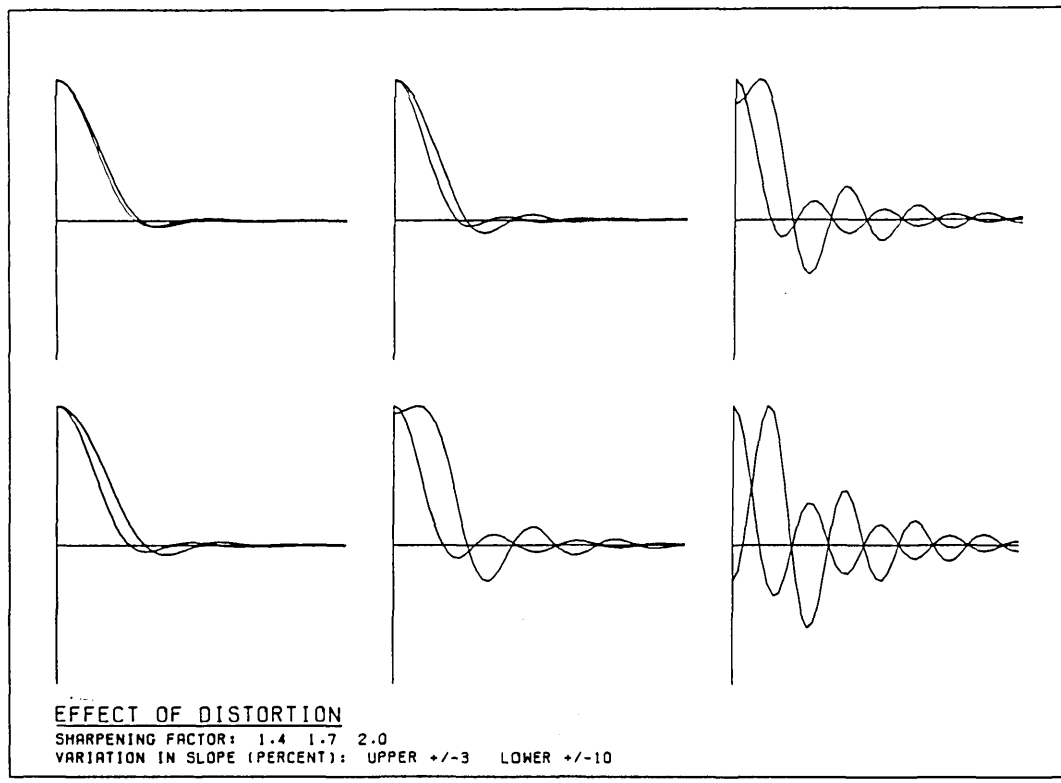


Fig.3.5f Effect of distortion on one dimensional deconvolution.

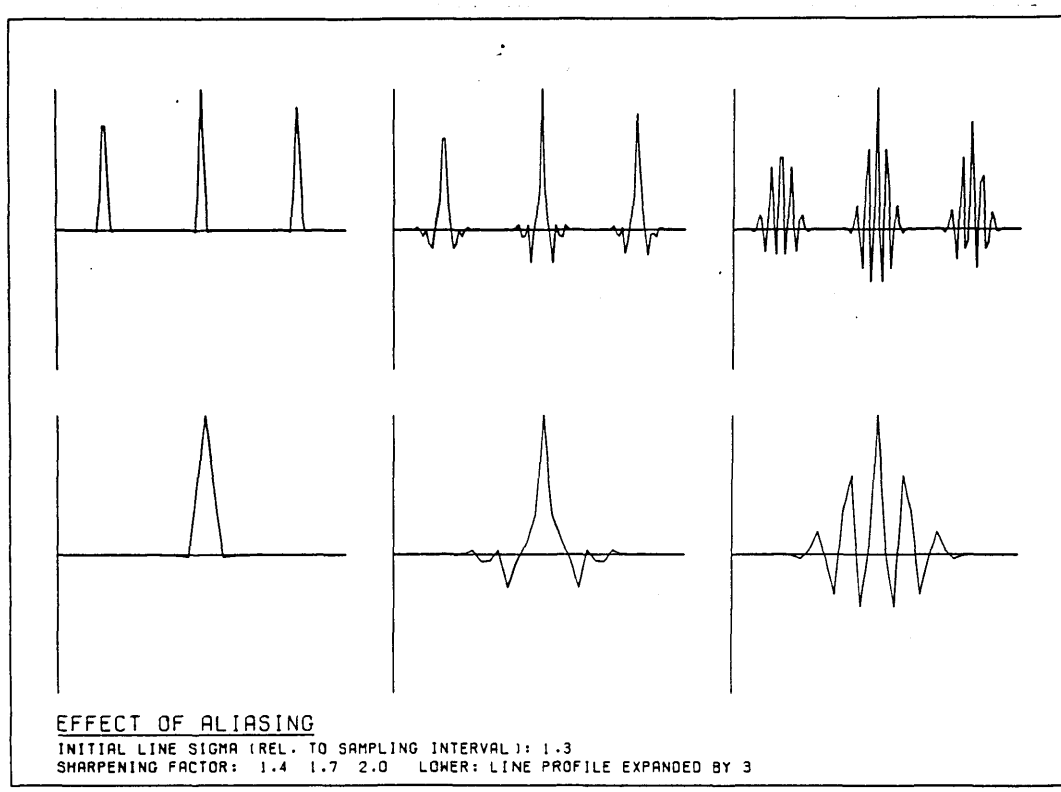


Fig.3.5g Effect of aliasing on one dimensional deconvolution.

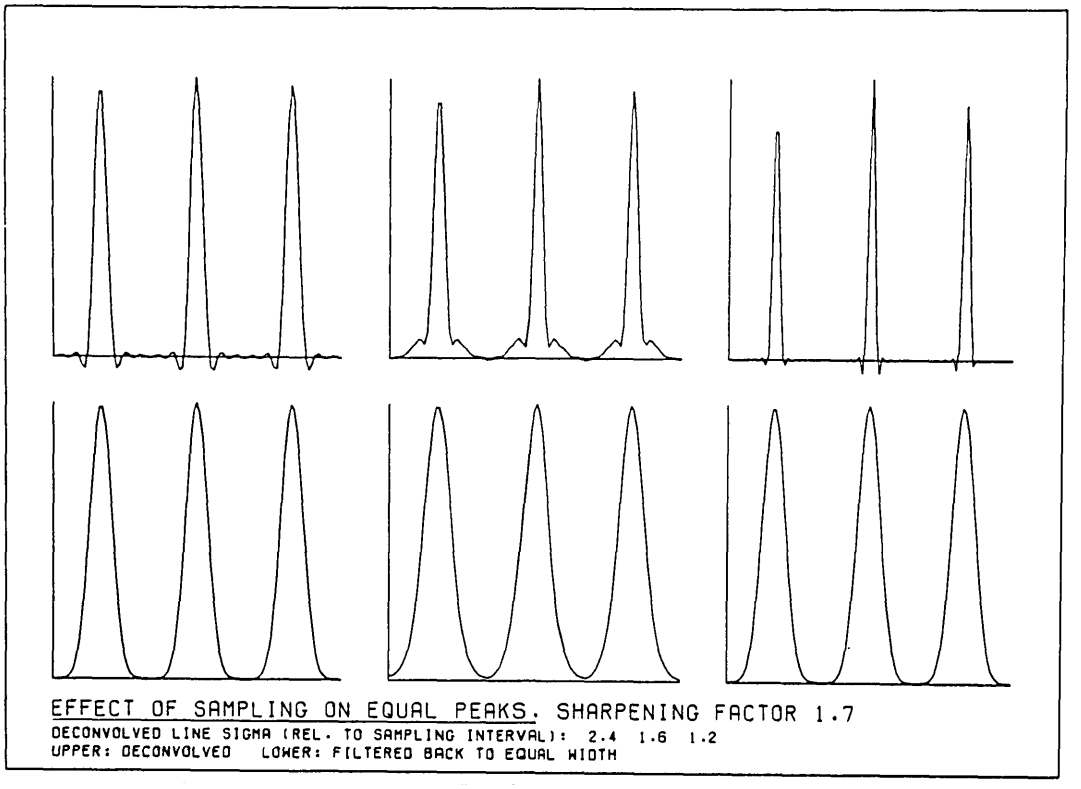


Fig.3.5h Effect of sampling on one dimensional deconvolution.

3.6 Deconvolution in Two Dimensions

3.6.1 Background

Deconvolution in two dimensions is precisely analogous to deconvolution in one. The principles of the method, the reasons for using it, and its properties and limitations are the same. In practice it is a little more complex, and the vastly greater amount of computation required makes efficiency a major consideration.

The method is similarly inapplicable when the frequency response falls rapidly to zero. This will arise if an image is badly out of focus or has moved during exposure. Correcting such defects, if possible at all, is more easily done with a maximum entropy method, and some degree of ambiguity in the restored image is inevitable. Diffraction limited images cannot be usefully sharpened, but are only likely to arise in optical astronomy when using space telescopes.

3.6.2 Method

For two dimensional images, the choice between convolutional and frequency space filtering is a matter of considerable importance. In general, convolutional filtering will be slower, and it will have the advantage only for filters with a small number of elements. The point of equal efficiency will depend on details of both the implementations and the machine, but typically it might be for a filter of 20 by 20 elements. It would be simple to improve the efficiency of the convolutional filter by writing in assembly language, which would increase this size to perhaps 40 by 40.

Frequency space filtering is however much more difficult to implement, as it requires a two dimensional fast fourier transform operating on the entire image. Added to the inherent complexity of programming and testing this transform, the limited memory available on the CDC would have forced the intermediate results to be held on a file and accessed out of sequence, leading to still more complexity.

Consequently convolutional filtering was chosen. The performance, though marginal with large filters, would be adequate at the sizes used. For small filters, representative of what might be used in practice for the moderate enhancement of large images, it would be the faster method as well as much the simpler.

To simplify the design and assessment of filters, circular symmetry was assumed, so that point images and filters could be described by their radial profiles. This is usually valid. Deviations, if not drastic, would not upset the results, but would need true two dimensional fourier transform and other routines to construct the filters.

The equivalent of a fourier transform in a problem with circular symmetry is the Hankel transform, in which the cosine is replaced by the zero order Bessel function. This was not implemented directly, but using the one dimensional fourier transform. The limitations of this were unfortunate. It involved linear interpolation of the input data in its conversion to a quadrant of a two dimensional image, with consequent approximations. Applied to filter frequency responses rising rapidly to very high values and then sharply cut off, it gave inaccurate filter profiles. This, and possibly other similar causes, made it impossible to construct filters giving a sharpening factor greater than about 1.5. Since it was felt that this covered most cases of practical interest, no improvements were made.

The deconvolved frequency responses used were, as for one dimension, rectangular, cosine, and cosine squared. The corresponding radial profiles show 13.2, 4.9, and 2.0% overshoot. These are shown in Fig.3.6a. In practice substantial overshoots, because they cover a larger area of the image than the central peak, are visually very obtrusive and confusing. About 5% seemed generally tolerable, which corresponded to the cosine squared frequency response when the filter was reduced to minimum size by windowing.

Typical filters for sharpening factors of 1.2 and 1.4 are shown in Fig.3.6b along with corresponding deconvolved profiles. Examples of deconvolved images are shown in Pl.3.6a. It is apparent that even sharpening by a factor of 1.4 can be a dramatic improvement in resolution.

3.6.3 Noise

In characterising the increase in noise produced by deconvolution filters, the sharpening factor has been defined in the same way as for one dimension, the zero dip criterion being applied to the profile along the line through the centres of the two image components. It is found that if the components are lines rather than points, the sharpening factor underestimates resolution by about 7%, and the overshoot is increased somewhat.

The deterioration in signal to noise for a point image and relative to a matched filter is shown in Fig.3.6c. It was found by numerical evaluation of about 40 filters, and the method was checked by using one on images. The results were also satisfactorily related to those for one dimension, and this was used to extend the curves from a sharpening factor of 1.5 to 1.6.

To indicate the extent to which the quality of astronomical images permits deconvolution, consider a star image formed at an f/15 cassegrain focus of a 3.5m telescope. This gives about the largest plate scale commonly used, and hence the best S/N. With the one arcsec seeing typical of a good site, the image is about 0.25mm across. For the finest grain plate in normal astronomical use, IIIa-J, and a peak image density of 0.5 above a 0.5 density background (which are reasonable restrictions to preserve adequate linearity), using Kodak data for the plate granularity gives a matched filter S/N of about 200.

Better S/N is obviously possible, for example by using a larger plate scale, but this suggests that a degradation by a factor of 10 or 20 is all that is tolerable on a normal good quality plate. The corresponding sharpening factor is 1.3 or 1.4. It would seem very exceptional that photographic images in optical astronomy would allow sharpening by more than 1.5, corresponding to a S/N degradation by more than 60. This is in agreement with the deconvolutional sharpening demonstrated by Arp and Lorre (1976), which they estimated as at best 50% for exposures on IIIa-J; they considered faster plates to be too noisy for the method to be used. Other instrumental methods, notably CCD sensors, could exceed this limit.

Slight sharpening, by 10% or so, degrades the S/N by a factor of only 3, and should be acceptable on any good plate.

3.6.4 Other Properties

The size of the filter has, for good efficiency, to be reduced to a minimum. It was found that a filter radius of 2.5 times the input profile sigma was adequate for sharpening factors up to 1.2, and 2.8 up to 1.4. These were achieved with careful adjustment of the width of the windowing function and the point at which it was truncated: the deconvolved profile does not deteriorate monotonically with decreasing filter length, and there is a series of optimum positions for the filter to be truncated. The possible saving on computer time justifies considerable care at this point.

Because only low sharpening factors were possible, little sensitivity to errors in assumed input profile width or shape was found. For a sharpening factor of 1.4 an error in width of 10% had little effect beyond the just tolerable alteration in overshoot by a factor of 2. Distortion, introduced by a transfer characteristic varying 10% from the mean slope, and appearing as clearly different shaped profiles, marginally altered both the overshoot and the sharpening factor. For practical use the limitation of the sharpening factor to about 1.5 by S/N degradation means there are no problems with sensitivity to input profile.

Sampling can cause aliasing as in one dimension. Much the same limit was found: there are no serious effects down to an input profile sigma in sampling intervals of 0.9 times the sharpening factor. Thus for a sharpening factor of 1.4, the filter can be as small as 9 by 9 elements if the sampling interval is correctly chosen.

3.6.5 Utility

The sharpening of two dimensional images by partial deconvolution has been demonstrated. A modest improvement in resolution, up to about 1.5, is obtainable without difficulty and adequately noise free images can be recorded by conventional photographic methods. The sharpening, while considerably less than the traditional limiting factor of 2, is enough to be very useful.

The method is little used, at least outside specialised labs processing images from interplanetary spacecraft. To a considerable extent its routine employment is prohibited by the scarcity of plate scanners and image processing systems, while the ease with which its misuse produces artifacts has discouraged its use even when possible.

There is a more fundamental restriction to its utilisation. The single possible major application in optical astronomy would be to reduce the the effects of atmospheric seeing, and the size of star images. Less dramatically than in the one dimensional case, the improvement is found to be illusory. Consider a sharpening factor of 1.4. The loss of S/N is by a factor of 23, so the deconvolved S/N would be equalled by the S/N of an unprocessed image taken with an exposure about 500 times shorter. Thus, an exposure of 40mins might be reduced to 5sec. And it is well known (and theoretically predicted) that while at a good site long exposure plates show a resolution of 1 arcsec, with exposures of a few seconds this falls to about ½ arcsec. (See, for example, Dollfus, 1961). Further, the seeing fluctuates from moment to moment, so that from a sequence of short exposures the best can be selected. Such high resolution is not confined to small telescopes: in the same volume as the above reference, ¼ arcsec resolution is claimed for a 1.5sec exposure of Jupiter on the Palomar 5m.

It is clearly easier, and more effective, to obtain better resolution directly than to use deconvolution on long exposure photographs. Higher sharpening factors are even less practicable: the object would have to be bright enough that it could be studied visually, when 0.15 arcsec resolution is frequently obtained.

Only for very slight sharpening might deconvolution be more effective. Taking as an approximate model an improvement in resolution by a factor of 2 as exposures are reduced from 5000 to 5 seconds and fitting a power law (power is 0.1), we would expect deconvolution to be about equally effective to a reduction in exposure time up to a sharpening factor of 1.2 (but no better), slightly worse at 1.3 and seriously so at 1.4. It is possible that until the exposure is reduced to perhaps a minute little improvement occurs, in which case deconvolution applied to long exposure plates would be the best way of sharpening by 1.2 or 1.3 for relatively faint objects.

The use of deconvolution to give any substantial gain in resolution would be of benefit only for the processing of pre-existing plates. In view of the unusually good S/N required for it to be possible, this must be a very minor role.

One possible use of a slight sharpening is apparent. If a scanner were to be used routinely to produce calibrated, linearised prints with standardised characteristics from original plates, the production of both normal and slightly sharpened prints would be a desirable if minor aid to interpretation. Or, on an equivalent interactive system, the user could be given an image sharpness adjustment that would extend from normal filtering continuously to slight enhancement. Such systems while quite feasible technologically currently lie considerably outside the financial constraints governing ground based optical astronomy.

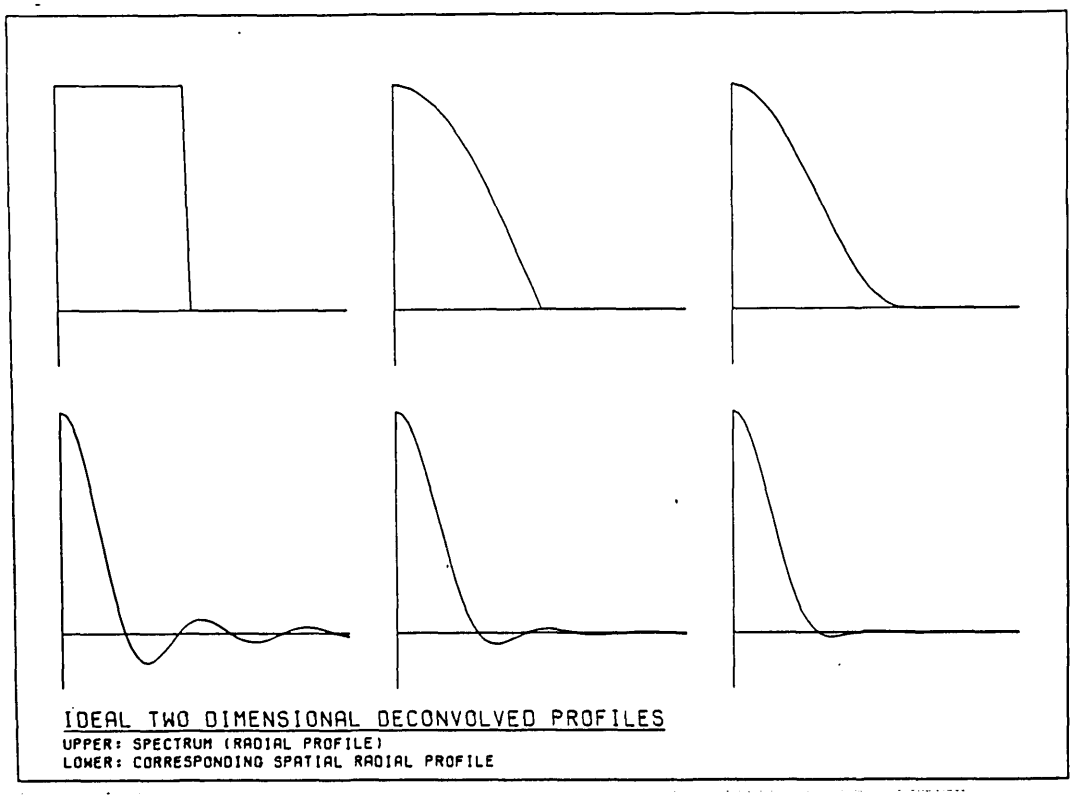


Fig.3.6a Ideal profiles produced by two dimensional deconvolution.

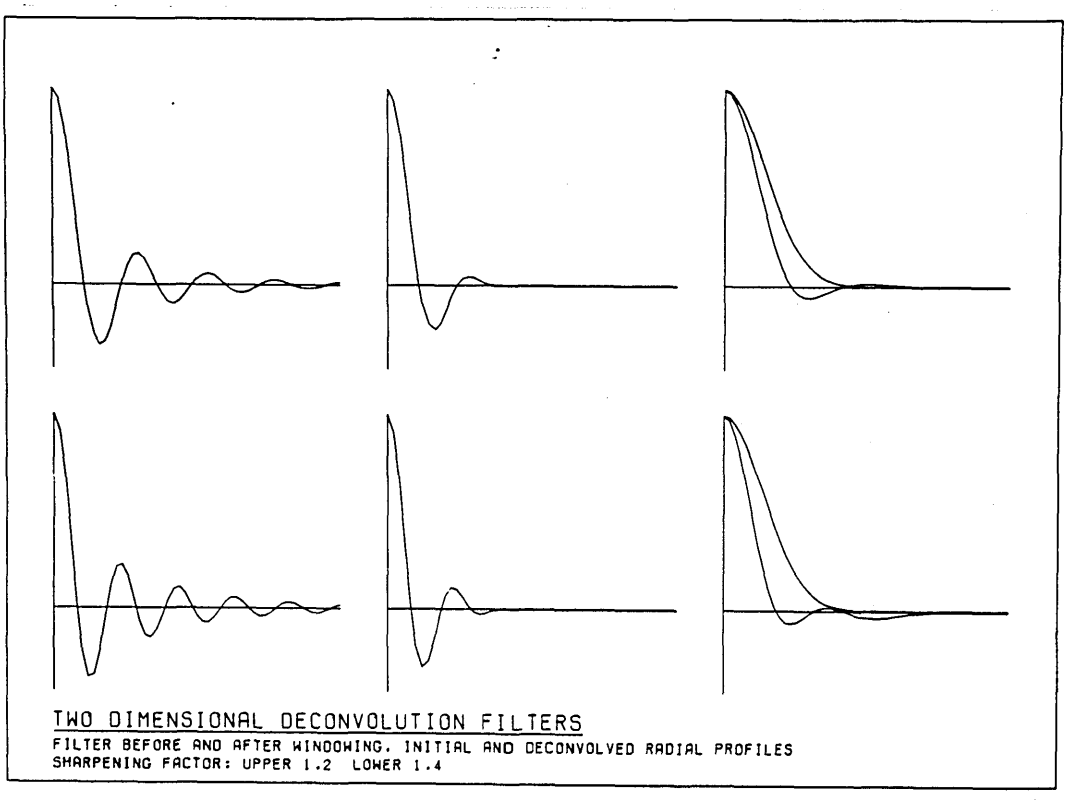


Fig.3.6b Two dimensional deconvolution filters.

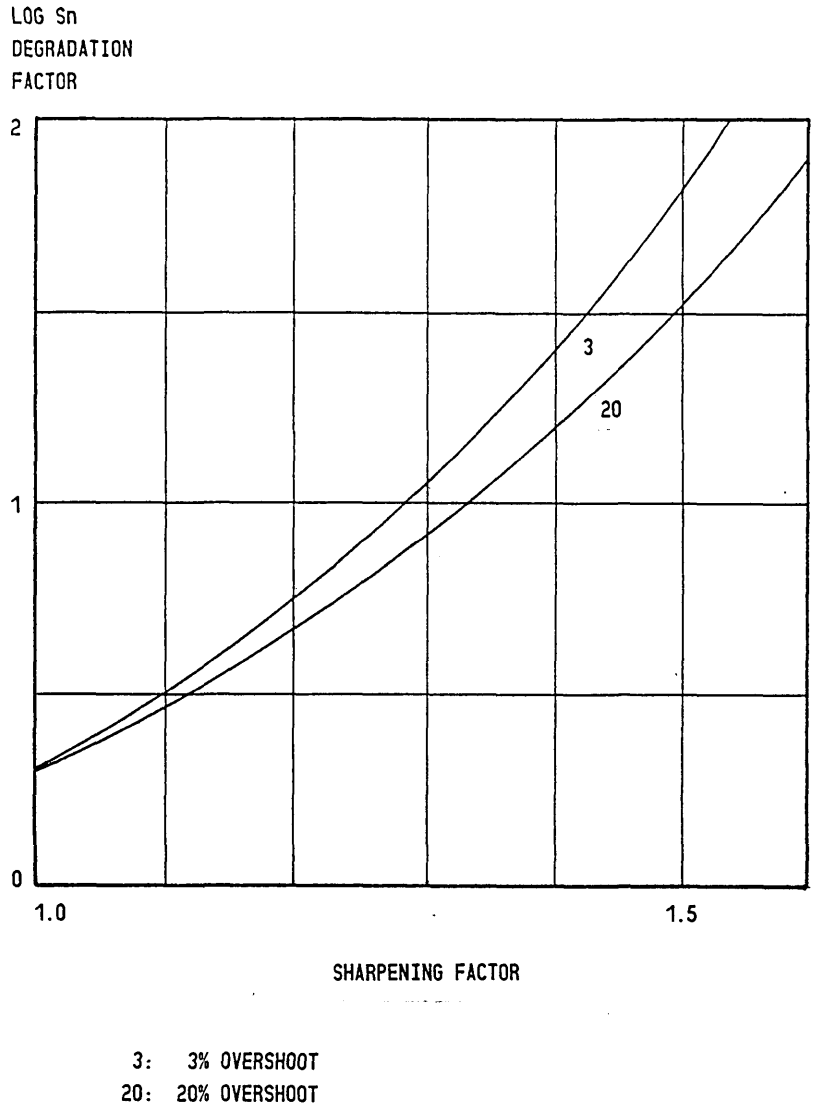


Fig.3.6c Signal to noise degradation in two dimensional deconvolution.
Accuracy: about 20% in degradation factor, less (10%) for 3% curve up to sharpening factor 1.4 (2 sigma).

3.7 Elliptical Galaxies

3.7.1 Introduction

Of all the types of astronomical objects that give larger than stellar images, elliptical galaxies form the most uniform group. Their shape is almost completely defined by a single parameter, the ellipticity. Variations in radial profile, and axial twist between the inner and outer regions, while detectable are not visually apparent. Their abundance is sufficient for statistical surveys to be useful, and so questions of detection limits and errors in classification are significant. They are simple enough to be amenable to the methods of study used for stars, while complex enough to be interesting, and so formed a logical extension of the previous work.

The de Vaucouleurs profile was used, for reasons indicated in 1.1.4. The scale parameter is the effective radius Re. The profile, normalised to unity integrated brightness is

$$(1/22.7Re^2)\exp(-7.67((R/Re)^{0.25}-1))$$

Note that this is sharply peaked at the centre. While exactly half the flux lies outside Re, the brightness at Re is only 0.05% of the central value. A table of the fraction of total flux within a given radius, calculated by numerical integration, is given for convenient reference as Table 3.7.

Nothing in this work would be expected to depend critically on the exact form of the radial profile. Observed profiles can be fitted adequately with the de Vaucouleurs model over the limited range of radius used, even if they deviate at extreme values. To some extent the results can be expected to apply, qualitatively at least, to disc galaxies. These show a narrow, bright spheroidal component, and a disc with a wider, exponential profile. But together these are not too dissimilar, over a limited range in radius, to a broader elliptical galaxy profile: this is evidenced by the uncertainty often experienced in trying to decompose the observed profile into components when data is lacking at very large and very small radii.

Ellipticity E is defined in the usual way, (1-(b/a)) where a,b are the major and minor axial scale lengths, the numerical class being 10 times this. The effective radius is Re. When the galaxy is not circular this is the geometric mean of the

values on the principal axes, which leaves the effective area and the central brightness unaffected by E.

Noise is assumed to have a gaussian distribution independent of the brightness at a point, which corresponds to a faint galaxy superimposed on a much brighter sky background. This adequately approximates the normal practical case. Even if the centre is much brighter than the background, the outer parts including all those significantly affected by noise will be fainter. The rms noise on a unit area is denoted An.

Other symbols used include the integrated brightness B and the widths (sigma) of the gaussian point spread function and a gaussian filter, Ws and Wf.

We also assume linearity. In practice this is only valid for very faint images. A normal deep plate, on a high contrast material such as IIIa-J, has a very small linear range and the centres of most galaxy images will be effectively saturated. This is not always a disadvantage, as it gives the image a more clearly visible shape. Where appropriate, images were produced using the limiting effect of the plotting routine in a similar way. The effects, when significant, will be noted.

3.7.2 Detection: Theory

The calculation of theoretical signal to noise ratios for an elliptical galaxy convolved first with a gaussian point spread function and then with a filter is analytically intractable and was done numerically. The results are shown in Fig.3.7a.

There is a trivial limiting case as Re tends to zero, and the galaxy approximates to a star image. The other limiting case is as Re tends to infinity; that is, as the effect of seeing becomes negligible. Each defines an asymptote to the actual plot of log(k) against log(Re), where k is the constant in $Sn=k(B/WsAn)$, Sn being the ratio of peak signal to rms noise, after filtering. Using a gaussian profile $Wf=Ws$ in the first and the de Vaucouleurs profile in the second case gives the best possible results, the filters being matched.

For perfect seeing, the matched filter gives $k=0.61(Ws/Re)$. Exact matching is not critical, and even for a uniformly weighted circular filter (assuming $E=0$) $k=0.42(Ws/Re)$ at the optimum radius of 0.16Re, or 68% of the maximum. A gaussian filter is substantially better: $k=0.51(Ws/Re)$ for $Wf=0.1Re$, or 84%. This is a negligible loss, and will be even less for a seeing degraded image. The width of the filter is not critical. For both the uniform circular and the gaussian it takes a width a factor of 4 off optimum (either way) to produce a loss of 20%, or ¼ magnitude, or a factor of 8 for ½ magnitude. This is a much wider tolerance than for star images, for which the factors are 2.0 and 2.8. Using a circular filter on an elliptical galaxy will introduce negligible loss, the inevitable mismatch in width on each axis being only by a factor of 1.8 even at $E=0.7$.

In practice a matched filter cannot be used. The wings of the filter extend inconveniently far even if outer parts contributing as much as 10% of the total weight are discarded, making for inefficiency and interference from adjacent objects. The filter profile would not be easily calculated, having a shape depending on Re/Ws. A gaussian filter is better in both respects, is a standard filter provided for general use, and allows the faster filter routine to be used: the loss of under 0.2 magnitudes is insignificant.

The performance with a gaussian filter on a seeing degraded image is plotted both for filter widths normally appropriate to preserve most of the detail of an image, Wf/Ws of ½ and 1, and for that giving the best Sn. Up to Re=10Ws the filter with Wf/Ws=1 is no more than ¼ magnitude less than the optimum. The difference increases at large Re, but even at 100Ws it is only ¾ magnitude. The optimum width is adequately approximated as Ws+(Re/10). The sensitivity to mismatch in width is about midway between the extreme stellar and galactic values at Re=15Ws.

3.7.3 Detection: Practice

Tests were made to confirm the calculated Sn, and to establish the total brightness required for a detected object to be seen to be non-stellar. Performance at large and small Re was measured separately. Over 300 images were produced, divided equally between continuous tone and contour plots, each field having 4 or 9 objects. The results are plotted in Fig.3.7b.

At large Re, from 5 to 40Ws, each image had a galaxy at E=0, 0.3, and 0.7, and a comparison star. They were plotted for noise levels at half magnitude intervals and for various width filters. Important images were repeated several times with noise derived from different seeds to improve the statistics. The maximum value of Re was limited by the plotting resolution and the need to keep the entire visual extent of the images within the plot.

Detection with an optimum gaussian filter agreed well with theory for a threshold of 4.5 times the rms noise. The best width of filter was not well determined, but seemed to agree with theory (2Ws at Re=10, 4Ws at Re=40), as did the effect of different widths. The narrow filter, Wf/Ws=½, made detection notably less efficient: at Re=40 it was equivalent to a drop in Sn by a factor of 2. This is noticably less than the theoretical figure of 3.0, showing the partial effectiveness of the eye as a filter. Detection was consistently slightly less effective for E=0.7 than for the less extreme shapes; the effective loss of Sn was about 20%. This is only marginally significant, but may reflect the apparent greater difficulty in recognising an elongated rather than compact peak when the orientation cannot be assumed. Results from continuous tone and contour plots were combined, there being no significant difference, and similarly for the two lower ellipticities.

Only a slightly greater level was needed for the object to be seen to be non-stellar. In all cases about 30%, or ⅓ magnitude, was required. There was no consistent dependence on either E or Re, and it was similarly insensitive to the width of the filter. Contour and continuous tone plots were again of similar effectiveness, except that contours were a little worse when E=0 and their shape could be no help.

It was noted that a higher signal level was required for the non-circular shape to be evident: perhaps 30% greater for E=0.7, and as expected still higher for E=0.3, by about a further 30%: but these are very tentative. Only for this, in the E=0.3 case, was there any major difference between contour and continuous tone plots. The latter were much worse than the above figures, requiring about double the signal level, and corresponding to the difficulty in seeing moderate ellipticities at the low radial gradient of a linear image.

It was found that it was very difficult to judge the size of a linear image in the presence of noise, the apparent size depending very largely on brightness. For equally conspicuous images (of moderate Sn around 50), the apparent size at Re/Ws=40 was only about 1½ times that at 10. Contour plots made judgement of size a little easier, but not much unless they were carefully measured.

At small Re, each plot had a series of galaxy images increasing in size by steps of 20%, and one star for comparison. These were made for E=0, 0.3 and 0.7, and with various noise levels. In all cases they were clearly visible: detection was not of interest. The filter used had Wf/Ws=½, as a reasonable compromise between resolution and noise.

The signal level required for the objects to be seen to be non-stellar was found, as expected, to increase rapidly with decreasing Re. At all levels, a variation by a factor of 1.7 in Re from the values plotted corresponded to marginal or obvious visibility, as opposed to positive discrimination with careful examination. To the Sn levels used, no difference was found between continuous tone and contour plots, though at very high levels contour plots must be superior.

These results, while correct as to general form, may be substantially in error, though probably by a factor of less than 1.5 in Re, because there were substantial inconsistencies within the data. The worst seemed due to a shift in the way discrimination was judged according to the apparent size of the other images in the field. They were aggravated by the difficulty in scoring a single image consistently. Better results could have been obtained by a much longer series of tests with randomised fields of galaxies mixed with stars, but with no immediate need for accurate values these were not done.

The determination of the threshold for visible ellipticity was even more erratic, but one main result was apparent. While at E=0.7 the ellipticity was apparent at Re little greater than needed for the object to be seen to be non-stellar, for either type of plot; at E=0.3 this was only true for contour plots: continuous tone plots required typically 2 or 3 times greater Re. This is another illustration of the difficulty of seeing slight ellipticity in the faint outer parts of images. But if the plot saturated, giving a sharp boundary round the peak, it was nearly as effective as proper contouring.

Together, the results show that there is maximum sensitivity for detection of an elliptical galaxy as non-stellar at about Re/Ws=3, requiring an integrated brightness about 5 times or 2 magnitudes greater than that needed to detect a star. Outside the range Re/Ws from 1 to 10, the total brightness neeeded increases rapidly. For large objects, it is proportional to their linear size. For small, the increase is much faster. About 5 magnitudes above the stellar limit is needed at Re/Ws=0.15. While not measured, the increase is even faster at smaller Re: very approximate theory indicates a further 5 magnitudes increase when Re is reduced by a factor of 3, and practical difficulties would make it worse than this.

Typical images used are given in Pl.3.7a and Fig.3.7c, which also serve to show the appearance of elliptical galaxies under various conditions.

3.7.4 Classification

For large and bright galaxies it is a simple matter to place them in their correct ellipticity class. The axial dimensions of an isophote large enough to be

little affected by seeing can be measured directly or the shape of the isophote can be estimated, or the shape of a saturated image can be used instead. But when, because of faintness or small Re, the size of the largest plottable contour or the apparent size of the image is little more than the size of a star image, the apparent ellipticity is less than the true value.

To investigate this effect, images were plotted, each with 9 galaxies of the same true ellipticity but with Re increasing by factors of 2. When noise was added the brightness was set so all the galaxies had the same Sn, the filter used being gaussian Wf/Ws=½ as the usual compromise between noise and resolution. For comparison, images were plotted each with a series of galaxies of increasing E but no noise, with slight or no filtering. A total of 84 plots were made, 31 continuous tone and the rest contour. Pl.3.7b and Fig.3.7d include a selection illustrating the main points.

The series against E showed a number of interesting features. It is not easy to see the ellipticity of linear continuous tone images when E is less than 0.2. Allowing the plot to saturate, giving a core with a sharp boundary, makes ellipticity visible down to at least 0.1: this is not entirely satisfactory with the plotting method used as it introduces a slight ellipticity of its own in this case. Contours are even better: E=0.1 is obvious, and 0.05 is just visible. Convolution with a gaussian (seeing or a filter) has negligible effect on the apparent ellipticity of linear continuous tone images if Re/W is at least 20; at 10 there is a slight, perhaps 10%, reduction in apparent ellipticity.

The appearance of highly elliptical galaxies, in these series and those without noise against Re, is notable. For E=0.7 or more the gaussian convolved image, over a wide range of Re/W, shows a relatively round core with wings extending to either side. This appearance is very marked at E=0.9. The image does not look at all elliptical, but seems to be made up of separate disc and bulge components, and looks very similar to many galaxies classed as S0.

The apparent ellipticity in seeing degraded linear images was found to have fallen to about half the true value when Re/W=0.7. Noise increased the discrepancy, especially at low E. At Sn=50 the 50% drop occurred at Re/W=1.0 for E=0.7, but at 7 for E=0.3, while in both cases Re/W=15 was required for negligible loss. The results are given in Fig.3.7e.

Contour plots, in the absence of noise, are not affected: a sufficiently low contour level can always be chosen. With noise, the reduction in ellipticity was found very similar to that for continuous tone images. The results were rather more erratic, being unduly affected by individual noise wiggles in the contour measured, but the drop was generally a little less.

3.7.5 Some Consequences

The variation with size of the threshold for detection of an object as non-stellar provides an alternative view of the selection effects noted by Arp (1966) and dicussed quantitatively by Disney (1976). Sensitivity is seen to be at a maximum for Re/W=3, and for a size greater or less by a factor of 4 there is a loss of ½ magnitude. For objects uniformly distributed through space, this corresponds to a reduction in the number detectable by a factor of 2, so a range in

size of little more than a decade is markedly favoured. The threshold rises rapidly at smaller sizes, indicating the severe selection effects for faint, distant, galaxies.

We also see that the total light from an elliptical galaxy of the most favourable size must be about 2 magnitudes above the threshold for detecting a star before the galaxy is detected as such. The exact value will depend on the profile; this assumes the de Vaucouleurs model. The apparent magnitude of faint galaxies will be seriously underestimated if assumed to be the same as comparably faint star images.

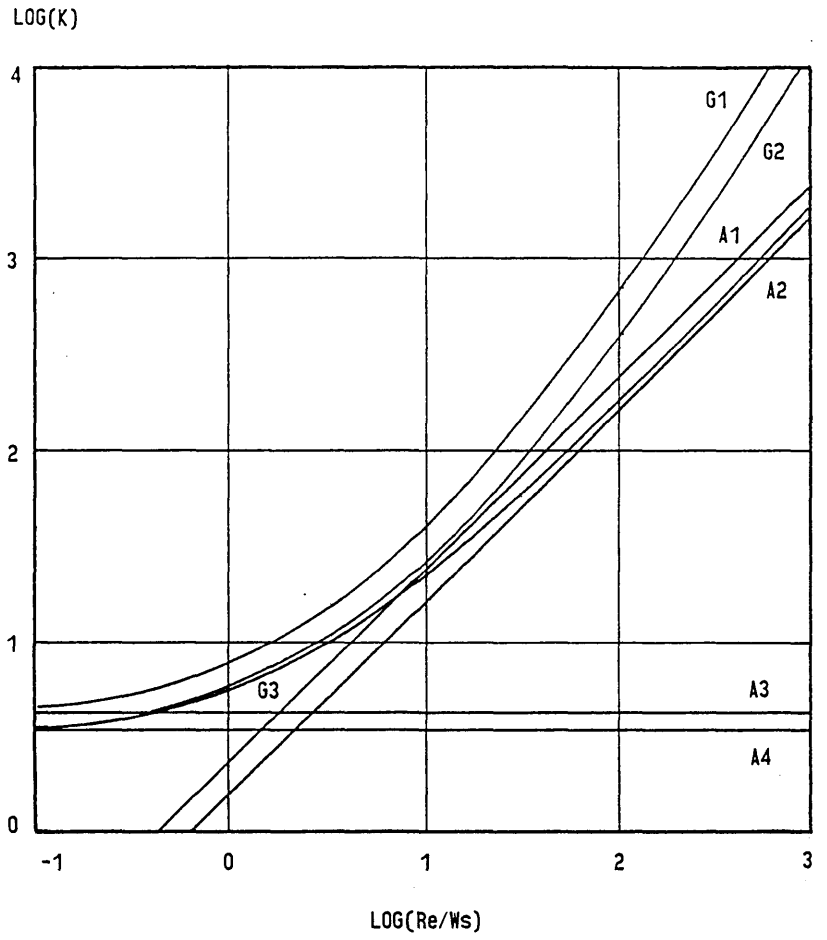
A typical galaxy has half the total light from inside the 22 mag per arcsec^2 isophote, and this is independent of size. Then for one arcsec seeing, if the galaxy is of optimum size for detection, the integrated brightness will be about 18 mag. If the sensitivity of a deep plate for stars is 25 mag, such a galaxy will be 5 mag above its detection limit. A smaller galaxy of the same surface brightness will be just detectable as non-stellar if it is about a sixth of the size, or Re is about ½Ws or ¼ arcsec. The detection threshold rises very rapidly at smaller Re, and slight changes in the assumed figures will not much alter this result. (Note that for such a typical galaxy, the diameter at 25 mag per arcsec^2, as used in the Second Reference Catalog, is about 6Re if the effect of seeing is ignored).

The difficulty in classifying small elliptical galaxies is indicated by the apparent loss of ellipticity at small Re. For large galaxies, Re/Ws of at least 10, there is no problem. For normal surface brightness galaxies, they are so far above threshold that there will be little error down to Re/Ws of perhaps 5, and this can be corrected if necessary. The centre of their image will normally be saturated, so that even judging by eye will be quite accurate. At smaller sizes the errors become large, and because the apparent size is almost entirely controlled by the brightness it is impossible to estimate Re adequately to allow correction of the apparent ellipticity. Correction is possible on a statistical basis, by assuming an average surface brightness, but it seems doubtful if reliable statistics can be obtained for the distribution among ellipticity classes for galaxies of Re/Ws much less than 5. For average surface brightness and one arcsec seeing this corresponds to about 17 magnitude.

There may also be difficulty in correctly recognising some small galaxies as elliptical. When the ellipticity is 0.7 or more they can appear much more like an S0 type. Unless the image is good enough that details of the disc or halo can be clearly seen, or that their absence is significant, the classification may be uncertain. This is largely avoided, except at much smaller sizes, by contour plots, which show outer isophotes of even highly elliptical galaxies as elliptical.

Log(R/Re)	%	Log(R/Re)	%	Log(R/Re)	%	Log(R/Re)	%
-2.0	0.4	-1.0	7.2	0.0	50.0	1.0	96.1
-1.9	0.5	-0.9	9.2	0.1	56.4	1.1	97.5
-1.8	0.7	-0.8	11.8	0.2	62.8	1.2	98.5
-1.7	1.0	-0.7	14.7	0.3	68.9	1.3	99.12
-1.6	1.3	-0.6	18.2	0.4	74.7	1.4	99.51
-1.5	1.8	-0.5	22.2	0.5	79.9	1.5	99.74
-1.4	2.4	-0.4	26.9	0.6	84.5	1.6	99.87
-1.3	3.2	-0.3	32.0	0.7	88.5	1.7	99.94
-1.2	4.3	-0.2	37.7	0.8	91.7	1.8	99.97
-1.1	5.6	-0.1	43.7	0.9	94.2	1.9	99.99
-1.0	7.2	-0.0	50.0	1.0	96.1	2.0	99.99

Table 3.7 Integrated flux as a function of radius for a deVaucouleurs profile.
Accuracy: possible error 1 in last place.



Sn=K(B/AnWs)

A1: GALAXY, NEGLECTING SEEING, OPTIMUM CIRCULAR FILTER
A2: GALAXY, NEGLECTING SEEING, MATCHED FILTER
A3: STAR, GAUSSIAN FILTER, Wf/Ws=0.5
A4: STAR, GAUSSIAN FILTER, Wf/Ws=1

G1: GALAXY, GAUSSIAN FILTER, Wf/Ws=0.5
G2: GALAXY, GAUSSIAN FILTER, Wf/Ws=1
G3: GALAXY, OPTIMUM GAUSSIAN FILTER

Fig.3.7a Theoretical detection sensitivity for elliptical galaxies.

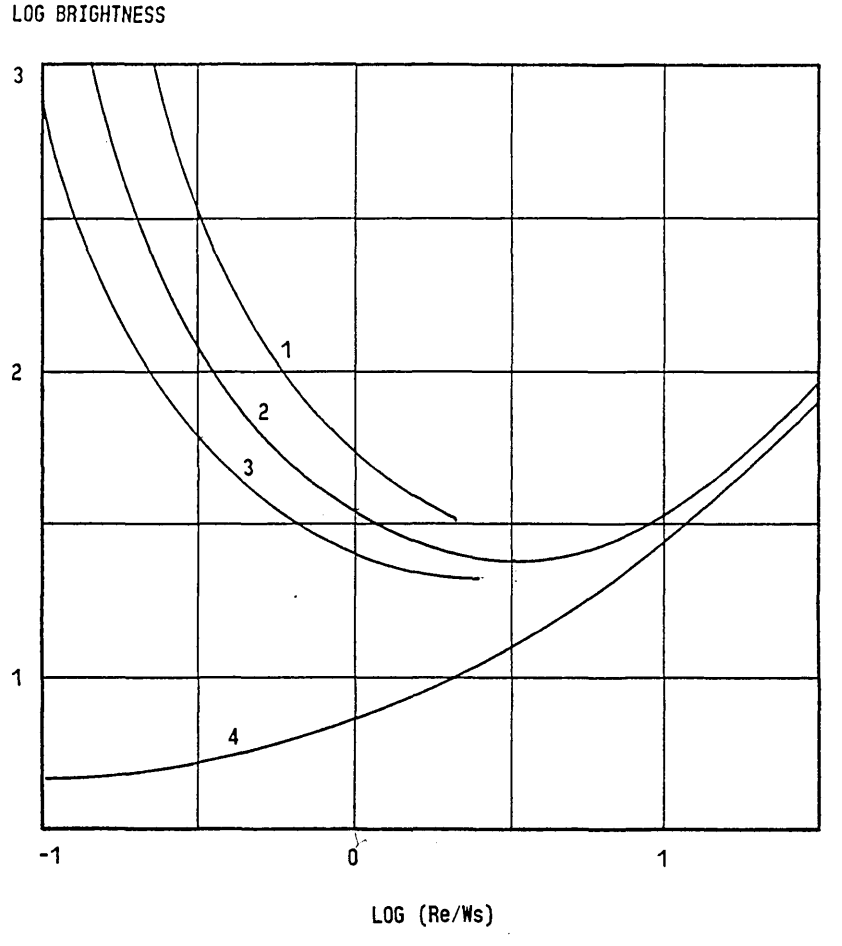


Fig.3.7b Actual detection sensitivity for elliptical galaxies.
Accuracy: errors are less than about 40% in Re on left branch, 20% in brightness on right (2 sigma).

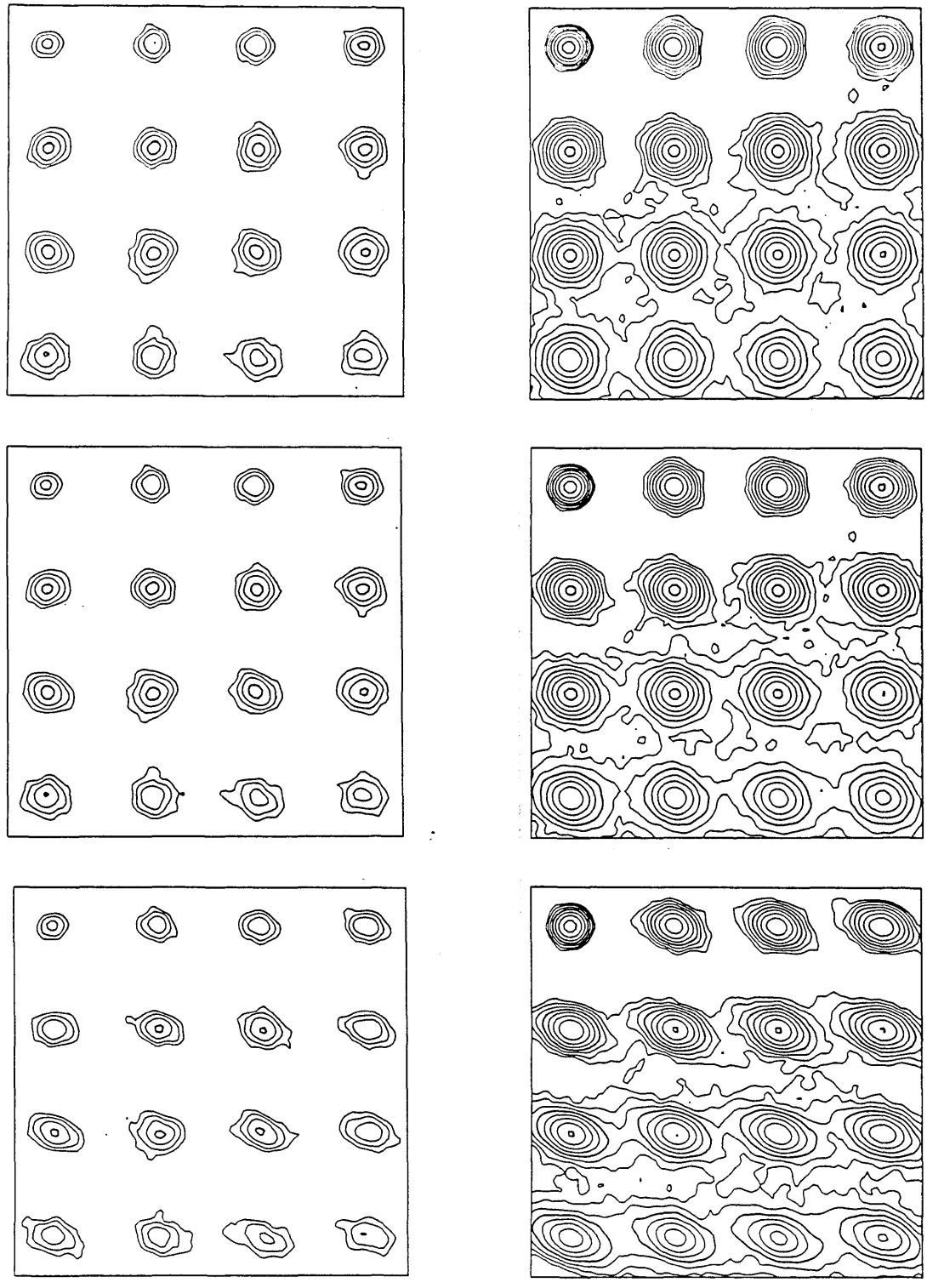


Fig.3.7c(i) Elliptical galaxies in contour plots: small Re.
Actual Sn approximately (l to r): 20, 130.
Ellipticity (t to b): 0.0, 0.3, 0.7.
Re/Ws: 0.44 to 5.0 by factors of 1.41. The first object is a star.
Gaussian filter Wf/Ws=½. Contours at ½ magnitude.

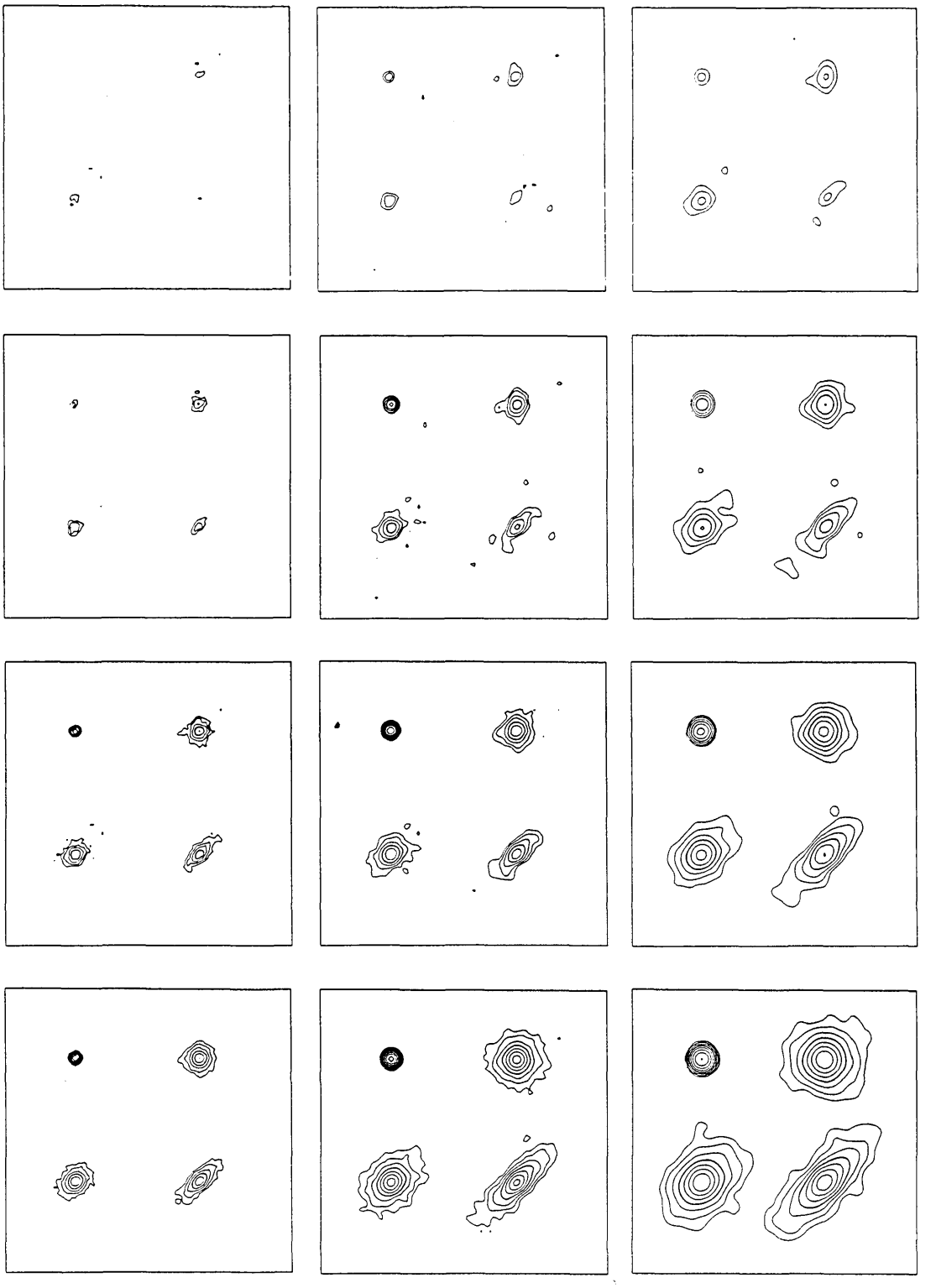


Fig.3.7c(ii) Elliptical galaxies in contour plots: large Re.
Each frame has a star and galaxies of ellipticity 0.0, 0.3 and 0.7, of approximately equal peak brightness after filtering. Brightness increases top to bottom in steps of 1 magnitude, starting at close to the detection threshold.
Filter gaussian, Wf/Ws (l to r): ½, 1, 2.
Re/Ws: 10. Contours at ½ magnitude.

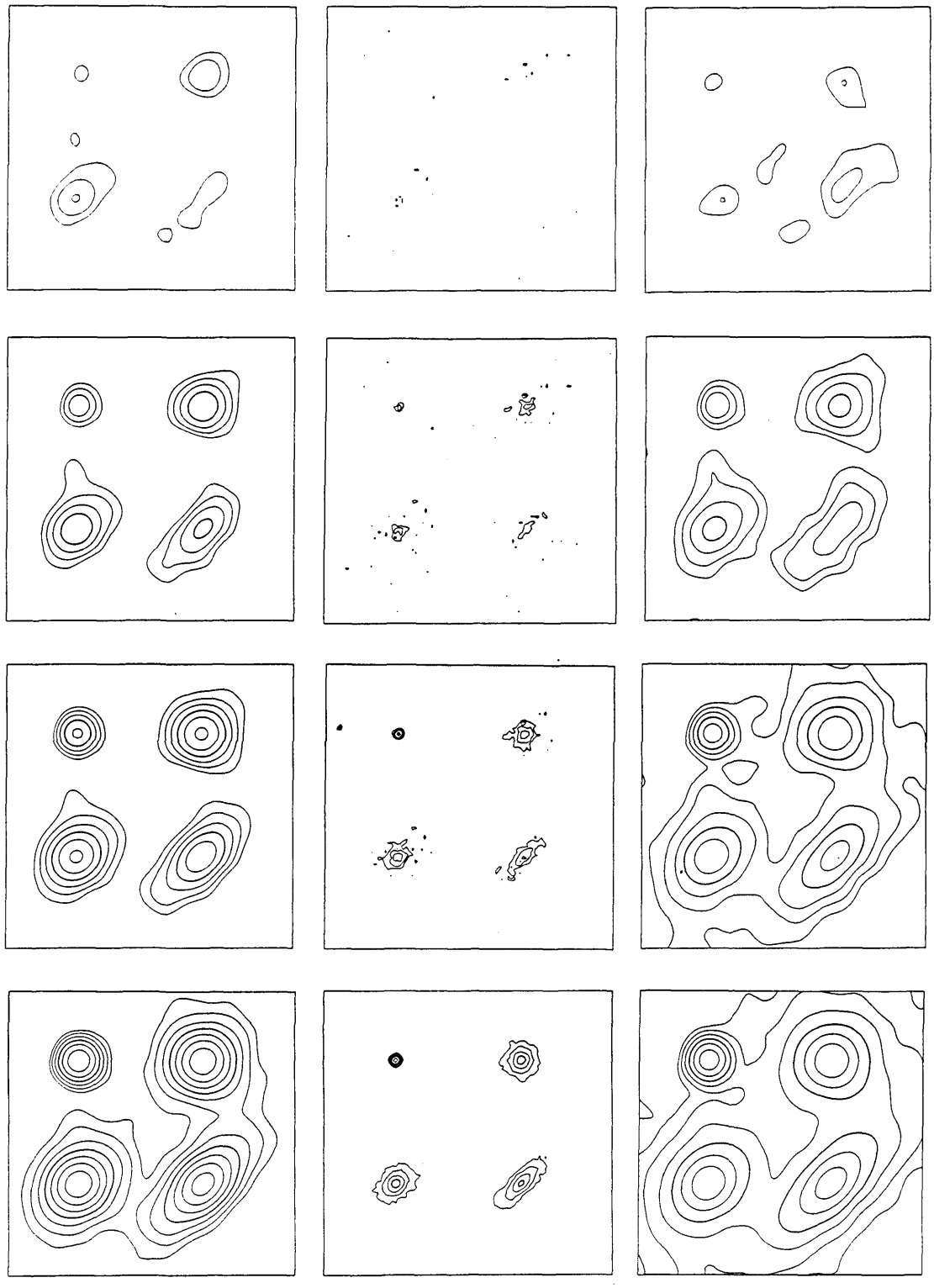


Fig.3.7c(iii) Elliptical galaxies in contour plots: large Re (cont).
Left: Re/Ws 10, Wf/Ws 4.
Centre: Re/Ws 40, Wf/Ws ½ (bottom frame is 1½ magnitude brighter than the one above).
Right: Re/Ws 40, Wf/Ws 4.

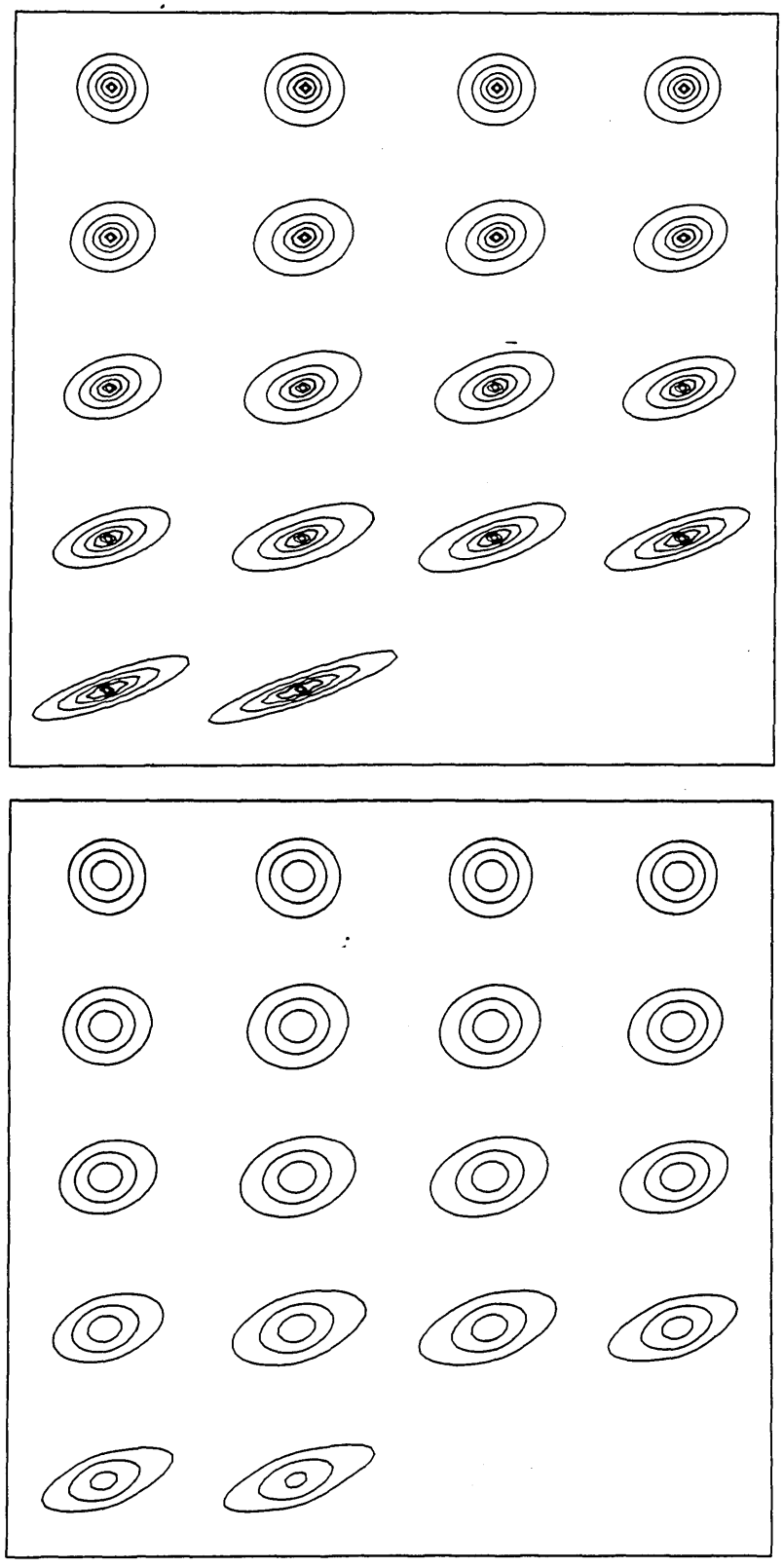


Fig.3.7d(i) Elliptical galaxies in contour plots: ellipticity.
Upper: unfiltered. Lower: filtered, gaussian Ws/Re=0.1.
Ellipticity: 0.0 to 0.9 by 0.05.
Re: 0.2 (relative to width of plot). Contours at ½ magnitude.

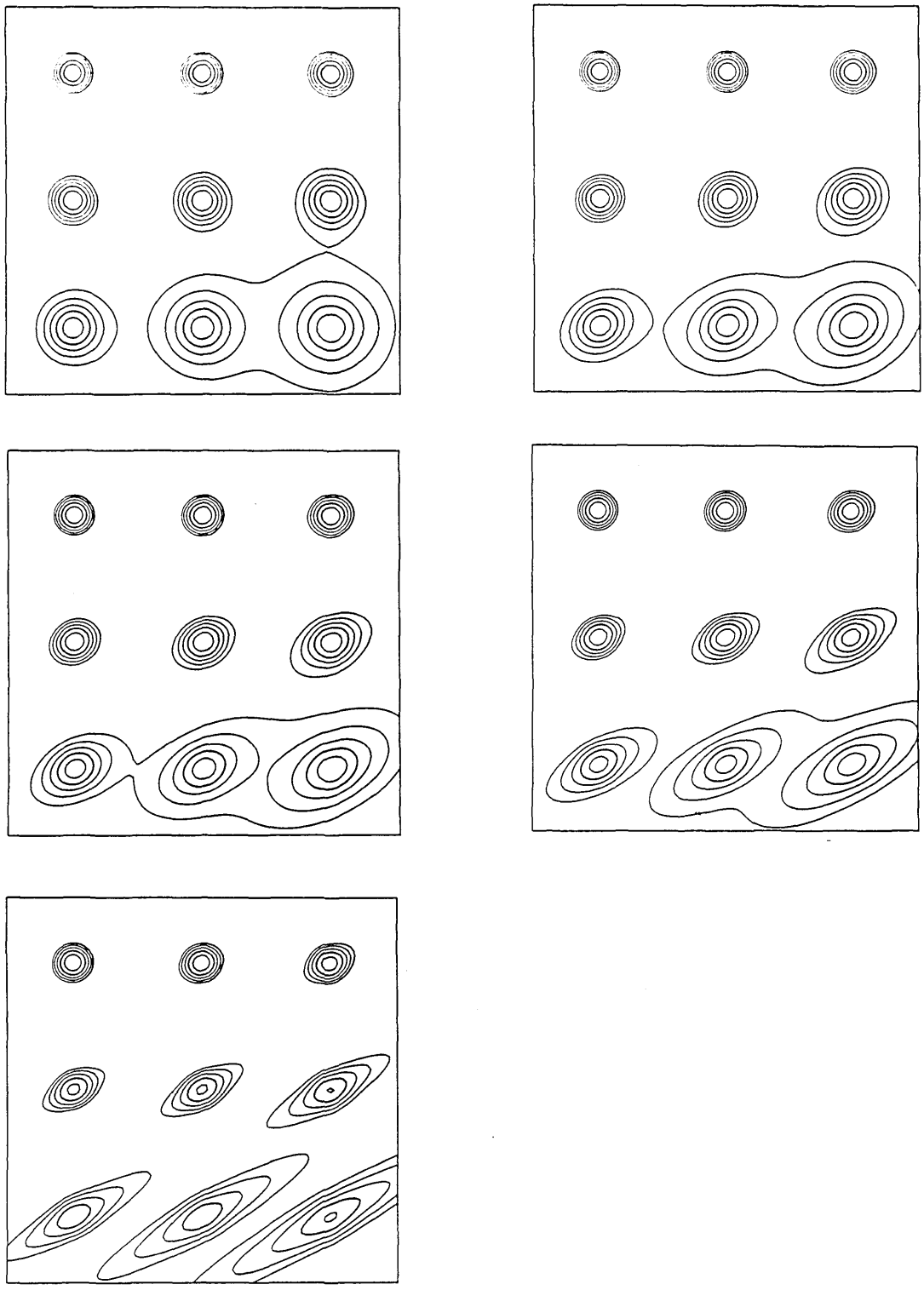


Fig.3.7d(ii) Elliptical galaxies in contour plots: Re.
Re/Ws: 0.07 to 18 by factors of 2.
Ellipticity: 0.0, 0.3, 0.5, 0.7, 0.9.
Convolved with a gaussian Ws=0.022 (relative to width of plot). Contours at ½ magnitude.

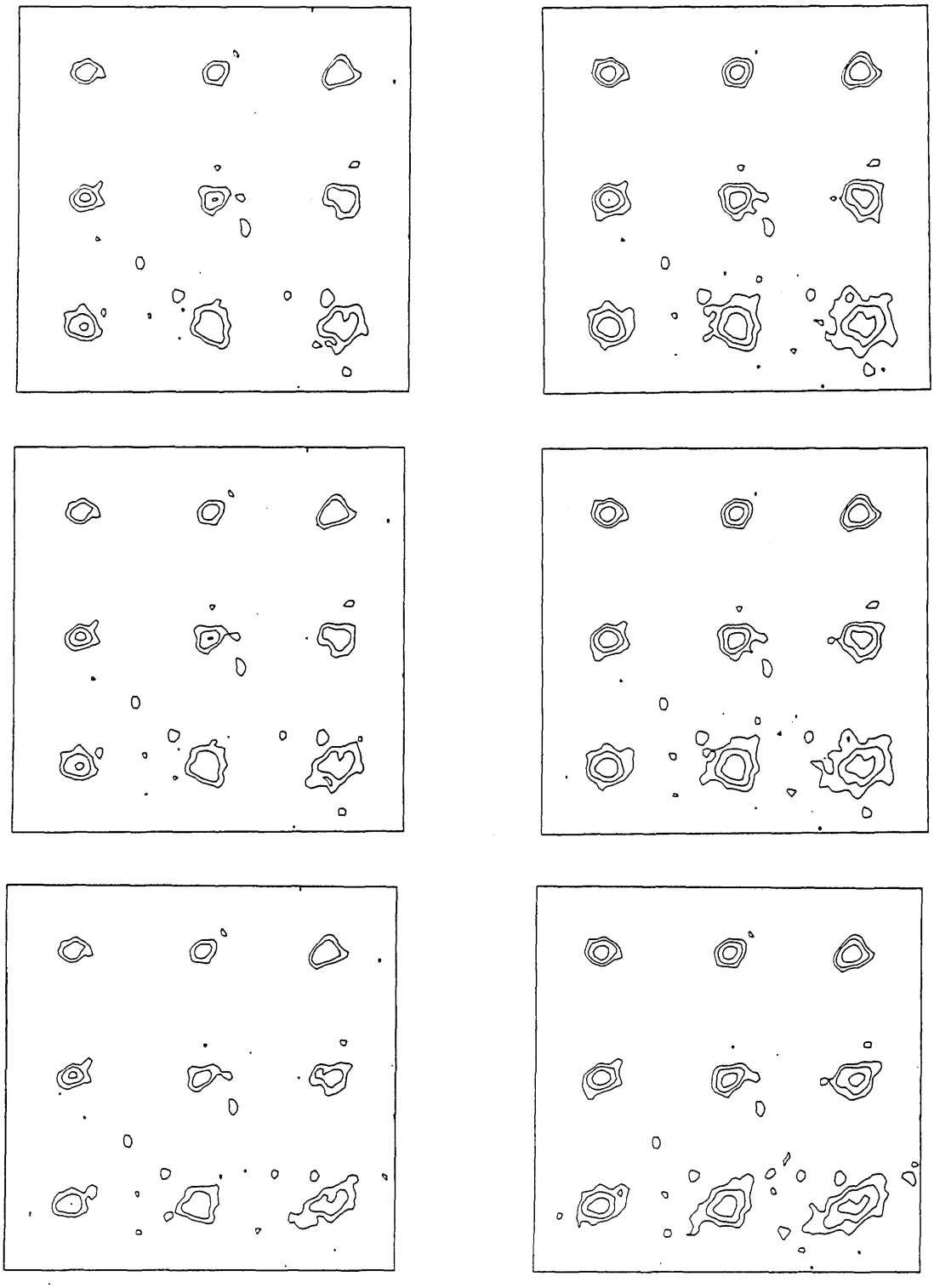


Fig.3.7d(iii) Elliptical galaxies in contour plots: with noise.
Re/Ws: 0.08 to 20 by factors of 2.
Ellipticity (t to b): 0.0, 0.3, 0.7.
Actual Sn approximately (l, r): 8, 12.
Convolved with a gaussian Ws=0.02 (relative to width of plot). Filtered with gaussian Wf=0.01. Contours at ½ magnitude.

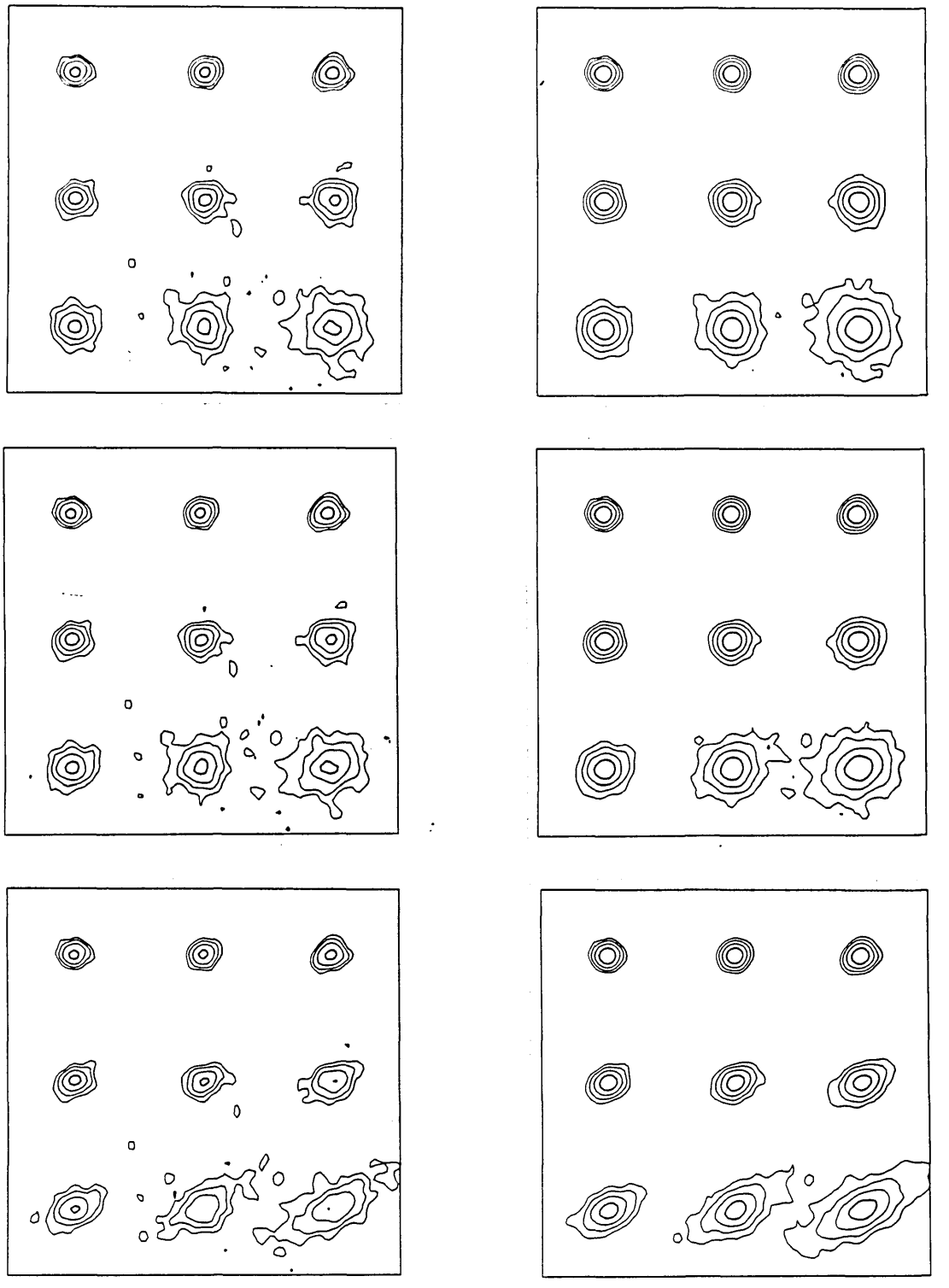


Fig.3.7d(iv) Elliptical galaxies in contour plots: with noise (cont).
Actual Sn approximately (l, r): 20, 50.

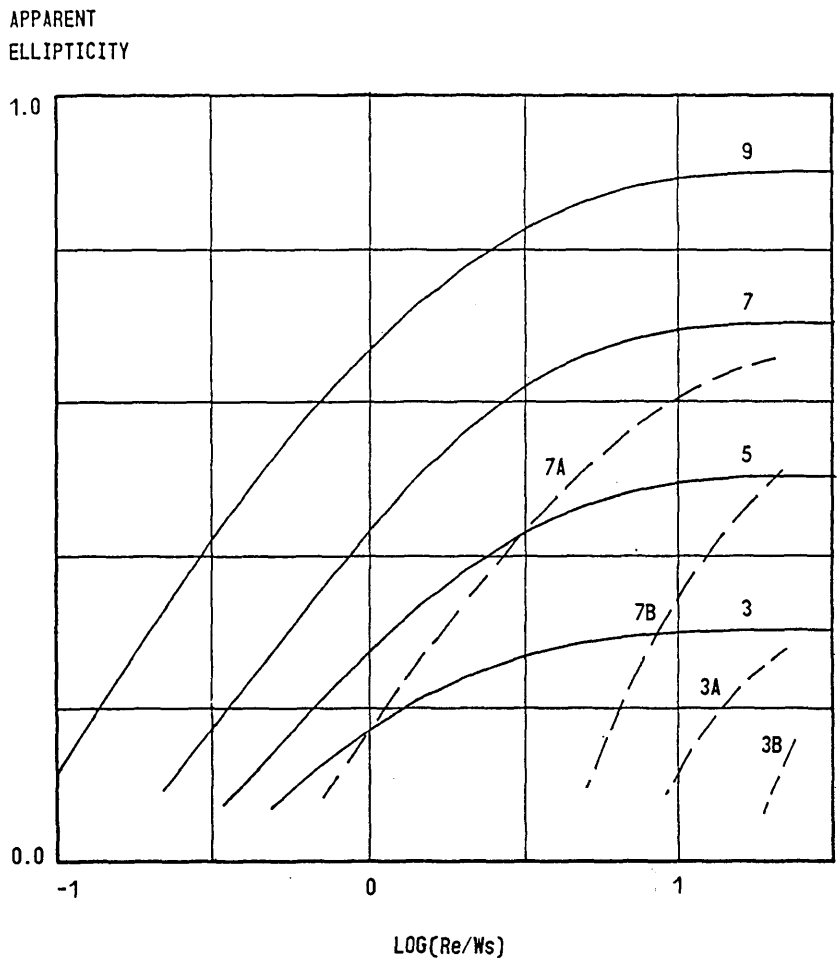


Fig.3.7e Apparent shape of elliptical galaxies.
For linear, continuous tone images.
Accuracy: apparent ellipticities are uncertain by 0.07, and worse at under 0.2 or with the higher noise level (2 sigma). Relative values are more accurate.

4.1 Summary

4.1.1 Computing

A simple but effective image processing system was implemented on a general purpose computer system, using a standard microfilm plotter as the output device. It is capable of producing both contour maps and continuous tone images. Resolution and efficiency are adequate for images up to at least 200 by 200 pixels: with minor changes 512 by 512 images could be handled. Some image manipulations are provided, chiefly two filters, one of which is capable of performing deconvolution.

The system is also capable of producing synthetic images. Fields of stars or elliptical galaxies can be set up and converted into images. Separate component images can be combined. Pseudorandom gaussian noise can be added to images and may be made dependent on image brightness so as to approximate to photon noise.

Another system was created to synthesise and manipulate one dimensional data. This is more specialised, and less general in its applications. It also provides fourier transforms, and the ability to handle circularly symmetric images described by a radial profile. It can produce images compatible with the processing system.

To provide the user interface to these systems, a general purpose command interpreter was written. A generator program enables a version to be created for a particular application, and also puts together the complete application program. These proved extremely flexible, and were indispensible in both program development and later use.

4.1.2 Results

The detection of peaks in noisy one dimensional data (and lines in spectra) and stars in fields of photon noise was demonstrated to closely follow theoretical predictions. Visual detection showed essentially full efficiency. Use of mismatched filters produced the expected reduction in performance, but this was quite slight even for filters of width in error by a factor of two. The loss was less than predicted for visual detection for narrower than optimum filters, the eye effectively providing

some of the missing filtering. Even for filters of grossly mismatched profile, having uniform weighting, the loss if they were of the correct width was usually negligible. The accuracy with which peaks could be located was also shown to agree with theory: smooth profile filters were shown to be needed.

Resolution of double peaks, lines and stars was investigated and the effects of noise measured. In all cases resolution was found to be at least as good as that indicated by the zero dip criterion, being in terms of the gaussian profile sigma about 2 for near equal components, and 3 at an intensity ratio of 30. Direct viewing of images was slightly superior to graphic or contour plots, at least at low ratios. Doubling was perceptible at smaller separations; only a little smaller for lines and one dimensional peaks, but down to 1 or less for double stars, the eye being very sensitive to slight deviations from the circular. For best results the filter had to be narrower than a matched filter, by a factor of 2 being generally suitable, but 4 when resolution was critical. For double stars displayed as contour plots, a uniformly weighted square filter resulted in unaccepatably irregular contours.

The improvement of resolution by partial deconvolution was tested in one and two dimensions. The deterioration produced in the signal to noise ratio was found to increase very rapidly when more than quite slight sharpening was attempted, and to be more if the ripples in the deconvolved spread function were minimised. Otherwise there were no difficulties in using the method. It seems improbable that improving the resolution by more than a factor of 1.7 in one dimensional data, or 1.5 in two, would ever be practical: these are rather less than the traditional factor of two. Even these modest gains were shown to be of no use in most circumstances, it being possible and more effective to obtain better resolution in the raw data, but deconvolutional sharpening was shown to suit a number of more specialised applications.

Detection of elliptical galaxies was shown to agree with theory, and the brightness required for them to be seen as non-stellar was determined. At best it was found that the galaxies had to be 2 magnitudes above the detection limit for stars to be seen as diffuse. At smaller sizes it was found much greater brightness was needed; at larger, the increase was slower. The apparent decrease of ellipticity at small size and low brightness was determined, and found to be potentially significant, and difficult to correct, for galaxies up to 10 times the minimum size normally discernable as diffuse. It was noted that highly elliptical galaxies could under some circumstances appear to have a near circular core and elongated wings, appearing not at all elliptical and more like an S0 type.

4.1.3 Continuation

There are four main lines on which this work could logically be continued to a limited degree. For any extensive continuation, it would be much preferable to transfer to more sophisticated image processing hardware.

The computer programs could be refined to make them easier to use and remove known weaknesses, and proper documentation written. They could thus be made suitable for general employment. Deconvolutional filters could be added in a form for routine use.

Numerical results could be refined, using the present data to plan the tests required. Reducing the uncertainty of the limits for detecting galaxies as elliptical would be particularly desirable, and so would obtaining the deconvolutional noise

increase against actual resolution increase for various intensity ratios, instead of that indicated by the zero dip criterion. Tests could also be repeated with different observers to find the variation between them.

Results could be checked against the observed performance on real astronomical data. This would be most useful, though difficult, for elliptical galaxies. Double star resolution and most of the results for spectra would be more amenable.

Studies of galaxies could easily be extended to other radial profiles, allowing generalisation to include spiral galaxies. It had been hoped to model more complex images, in particular to study the appearance of S0 galaxies and their discrimination from ellipticals. This would also have made it possible to look at the appearance of randomly superimposed galaxy pairs, to see the frequency with which they gave an illusory appearance of interaction, and the extent to which this increases when photographic non-linearity is included.

4.2
References

The references given are mostly to the sources of data used for comparison or discussion. None were found which were directly relevant to the results obtained, but no systematic search was conducted and the older literature was largely neglected.

Image processing now has a vast literature, of which very little is specifically concerned with astronomical applications. While it was consulted for background information, it was not directly used and is not cited.

Arp, H. 1966: 'Atlas of Peculiar Galaxies' (Pasadena)
Arp, H. and Lorre, J. 1976: ApJ, 210:58
Conrady, A. E. 1929: 'Applied Optics and Optical Design' (Oxford)
Disney, M. J. 1976: Nature, 263:573
Dollfus, A. 1961: in 'The Solar System', Vol III, ed. Kuiper, G. P. and Middlehurst, B. M. (Chicago)
Glasby, J. S. 1968: 'Variable Stars' (Constable)
Kodak 1971: Data Sheet SE-3, 'Kodak Scientific Films and Plates'
Sawyer, R. A. 1944: 'Experimental Spectroscopy' (Prentice-Hall)
Stock, J. and Williams, A. D. 1962: in 'Stars and Stellar Systems', Vol II, ed. Hiltner, W. A. (Chicago)
Strand, K. Aa. 1963: in 'Stars and Stellar Systems', Vol III, ed. Strand, K. Aa. (Chicago)
Smith, W. J. 1966: 'Modern Optical Engineering' (McGraw-Hill)
Walsh, J. W. T. 1958: 'Photometry' (Constable)

A1 Programs: Plotting

A1.1 Modes and Options

Four integer parameters, used throughout the system, select the plotting and layout modes required and various minor options. The more important options are given along with the corresponding values. Where more than one set of values are given for a parameter, one choice from each set is made and the values added.

MO: operating mode

This selects the basic type of plot required.

0, 200, 500, 600 :printer, numeric dump, continuous tone, contours

The last two digits select the method used if the data has to be re-sampled at a different size grid.

MP: picture format

This controls the layout on the paper or microfilm, and has a different meaning in each case.

printer:

0, 100, 200, 300 :printer lines per inch: 6, 6.25, 8, 8.25

microfilm:

0, 1000, 2000 :frame area: full, standard 35mm, within 35mm mask

0, 200, 400, 600, 800 :grey scale: none, steps, wedge, steps (labelled), wedge (labelled)

0, 100 :partial frame border: off, on

0–99 :plot scale; the number of pixels to the full width of the frame divided by 20

MB: border to image

This controls the details of the border round the image.

0, 1000 :full, sides only

0, 20, 220, 425, 285 :full, none, corner dots, standard, standard (contours)

0, 1, 2, 3 :width in pixels: 0, 1, 3, 9

MF

This controls the options of processing by sections, and the types of labels and headers, as well as modifying printer output. Only the parts of importance to basic use are given.

0, 2000 :printer output to: unit 10, unit 9

0, 1000 :printer output: paged, continuous

0, 4 :labels: none, full

A1.2 Common Variables

The common blocks are listed and the more important parameters in them.

COMMON/STATUS/IEQ,IER,IES(32)

IEQ	:error handling, 0=return, 1=call ERROR
IER	:set to value greater than 0 to indicate error
IES(32)	:limits and status, only 3–16, 30–32 used
4, 5	:last cp time in seconds, its limit
6, 7	:microfilm max pages, max plotfile length
8	:intensity last set
9, 10	:lines output to unit 9, 10
11, 12	:microfilm pages output, file length
13–16	:two current x,y positions
30–32	:limiting data value, two constants used in data scaling

COMMON/KON/K(128),KONLEV(128),KONSYM(128),KONTAB(2048)

K(128)	:the preset parameters for controlling plotting a point in printer, contour and continuous tone modes. Only used up to 71 (and that only partly).
2, 3	:data scaling, origin and range
6–12	:printer
6–8	:expansion, base level, interval
9–11	:switches; log, table lookup, invert data (0/1=off/on)
12	:number of overprints
23–35	:contouring
23–25	:expansion, base level, interval
26, 27	:switches; log, table lookup (0/1=off/on)
28, 32	:emphasis interval, max plotted intersections
33–35	:intensities; symbols, faint contours, bright contours
41–71	:continuous tone
41–43	:expansion, base level, interval (normally 1)
44–46	:switches; log, table lookup, invert data (0/1=off/on)
47	:type of plotting symbol (4 is a skewed line)
50	:intensity of border symbol
51, 52	:second segment break point
53	:slope parameter (typically 256)
55–59, 61–65, 67–71	:parameters for each of the 3 ranges. For each; line intensity, lower threshold, scaling divisor, transition parameters (2)
KONLEV(128)	:table of contour levels corresponding to the table in KONTAB (not used)
KONSYM(128)	:symbol table for line printer output (not used)
KONTAB(2048)	:lookup table for converting data (not used)

COMMON/PROCOM/IP(30)
COMMON/LENGTH/LEN(4)
These hold the preset parameters and the index variables controlling the plotting operation.

IP

1–7	iblk kblk lblk mblk nblk nbblk nlpblk These govern the allocation of lines into blocks, iblk indicating the current block of nblk (kblk being an offset) and mblk selecting the block(s) to be processed, nbblk and nlpblk giving the number of border and overlap lines.
8–14	isec ksec lsec msec nsec nb nlap Similarly, these govern subdivision along lines into sections.
15–20	idl iol ipl ndl nol npl These are indicies to and numbers of data lines, output image lines, and plotting lines.
21–25	mb mf mo mp ms These are the mode parameters, ms indicating the input to be used.
26–28	iil ilap kpl Indicies to interpolated line position and overlap line, and offset along plotting line.
29, 30	kida kidb Two identity numbers written to the plot.

LEN

1–4	ldl lol lpl lppl Length of data lines, output image lines, plotting lines, and length of printer line.

A1.3 Routines

Routines are listed by name only, and then with brief descriptive notes. The source code is then given. They are arranged:

Initialising	PLOTZ to SETSYM
Exectutive	PRORDY to LNPLOT
Printer	LPRTZ to LDMP
Microfilm	MFZ to MBOX

```
SUBROUTINE PLOTZ  (LL,NL,KBASE,KRANGE,KIDA,KIDB,MS,NGS)
SUBROUTINE PLOTC  (LL,NL,KBASE,KRANGE,KIDA,KIDB,MS,NGS,KLOW)
SUBROUTINE SETZ   (LL,NL,KBASE,KRANGE,KIDA,KIDB,MS,NGS)
SUBROUTINE SETC   (LL,NL,KBASE,KRANGE,KIDA,KIDB,MS,NGS,KLOW)

SUBROUTINE SETBOR (MB,NB,NBBLK,NLAP,NLPBLK)
SUBROUTINE SETBOX
SUBROUTINE SETDIM (LDL,LOL,LPL,LPPL,KPL,NDL,NOL,NPL)
SUBROUTINE SETLAB (KIDA,KIDB)
SUBROUTINE SETLOG
SUBROUTINE SETMOP (MF,MO,MP,MS,KIDA,KIDB)
SUBROUTINE SETSEC (LBLK,MBLK,NBLK,LSEC,MSEC,NSEC,KBLK,KSEC)

SUBROUTINE SETPRZ
SUBROUTINE SETCOM (MKON,MBOP,MSDIM,KBASE,KRANGE,KIDA,KIDB)
SUBROUTINE SETKON (M,KB,KR)
SUBROUTINE SETPRO (MBOP,MSDIM,KIDA,KIDB)
SUBROUTINE SETVAL (I,N)
SUBROUTINE SETLIM (ITIME,IPAGE,IFILE,IQ)

SUBROUTINE SETKSD (M,L,NA,NB,NC,ND,NE,NF,NG,NH)
SUBROUTINE SETKSX (J,L,IS,ID,IL)
SUBROUTINE SETPSD (M,L,NA,NB,NC,ND,NE,NF,NG,NH)
SUBROUTINE SETSYM (IB,IS,NL,NM,ITAB,IALT)

SUBROUTINE PRORDY (ML,LDL,LPL,LPPL,JA,JB,JD,JP)
SUBROUTINE PROSEQ (ML)
SUBROUTINE PROLN  (ML,LDL,LOL,LPL,JA,JB,JD,JO,JP,JQ)
SUBROUTINE EXPLOT
SUBROUTINE NEWLN  (MSS,MA,MX,J,L)
SUBROUTINE BORDER (NB,NLAP,KSYM,MB,IOL,IIL,ILAP)
SUBROUTINE ERROR
SUBROUTINE BLKEND (ISEC,NSEC,IBLK,NBLK,KIDA,KIDB,MF,ML,MO,JP,LPL)

SUBROUTINE LNSPEC (IIL,IOL,NOL,NDL,MO)
SUBROUTINE LNDATA (JA,JB,JD,LDL,IIL,MO,MS)
SUBROUTINE LNSCAL (JO,LOL,JD,LDL,MO)
SUBROUTINE LNBORD (JP,LPL,LOL,KSEC,LSEC,MSEC,NSEC,MB,MF,ML,KPL)
SUBROUTINE LNOUT  (JP,LPL,JO,LOL,KSEC,LSEC,MSEC,NSEC,MB,MF,IOL,IIL,ILAP,KPL)
SUBROUTINE LNPLOT (JP,JQ,LPL,MF,MO,MP)

SUBROUTINE LPRTZ  (JP,LPL,MF)
SUBROUTINE LPRT   (JP,LPL,MF,NP)
SUBROUTINE LMERGE (JP,LPL)
SUBROUTINE LDMP   (JP,LPL,MF)

SUBROUTINE MFZ    (JP,LPL,MP)
SUBROUTINE MFC    (JP,JQ,LPL,MP)
SUBROUTINE MFCL   (IP,IQ,KX,KY,IX,IY,N)
SUBROUTINE MFPAGE (MF,MO,MP,LPL)
SUBROUTINE MFGS   (MO,MP)
SUBROUTINE MFEND  (ISEC,NSEC,IBLK,NBLK,KIDA,KIDB,K)
SUBROUTINE MFSYM  (KX,KY,IX,IY,IA,IT,M,N)
SUBROUTINE MBX    (KL,KR,KB,KT,IT)
```

```
SUBROUTINE PLOTZ (LL,NL,KBASE,KRANGE,KIDA,KIDB,MS,NGS)
  PLOTS FROM FILE MS ONTO MICROFILM.

SUBROUTINE PLOTC (LL,NL,KBASE,KRANGE,KIDA,KIDB,MS,NGS,KLOW)
  PLOTS LOG CONTOURS FROM FILE MS ONTO MICROFILM.

SUBROUTINE SETZ (LL,NL,KBASE,KRANGE,KIDA,KIDB,MS,NGS)
  PREPARES TO PLOT FROM MS (0 DEFAULTING TO TAPE7) ON MICROFILM.

SUBROUTINE SETC (LL,NL,KBASE,KRANGE,KIDA,KIDB,MS,NGS,KLOW)
  PREPARES TO PLOT FROM MS (0 DEFAULTING TO TAPE7) AS CONTOURS
  ON MICROFILM.

SUBROUTINE SETBOR (MB,NB,NBBLK,NLAP,NLPBLK)
  SETS THE BORDER MODE AND PARAMETERS

SUBROUTINE SETBOX
  SETS BOX BORDER

SUBROUTINE SETDIM (LDL,LOL,LPL,LPPL,KPL,NDL,NOL,NPL)
  SETS THE LINE LENGTHS AND THE NUMBERS OF LINES

SUBROUTINE SETLAB (KIDA,KIDB)
  SETS THE IDENTITY NUMBERS

SUBROUTINE SETLOG
  SETS LOG SWITCHES.

SUBROUTINE SETMOP (MF,MO,MP,MS,KIDA,KIDB)
  SETS THE MODES (EXCEPT MB) AND IDENTITY NUMBERS

SUBROUTINE SETSEC (LBLK,MBLK,NBLK,LSEC,MSEC,NSEC,KBLK,KSEC)
  SETS THE SECTIONING PARAMETERS

SUBROUTINE SETPRZ
  SETS PROCOM TO ZERO

SUBROUTINE SETCOM (MKON,MBOP,MSDIM,KBASE,KRANGE,KIDA,KIDB)
  SETS ALL THE COMMON ARRAYS

SUBROUTINE SETKON (M,KB,KR)
  SETS KON (K AND KSYM) WITH A SET OF VALUES.

SUBROUTINE SETPRO (MBOP,MSDIM,KIDA,KIDB)
  SETS PROCOM WITH A SET OF VALUES
  THIS VERSION IS A DUMMY

SUBROUTINE SETVAL (I,N)
  SETS A SINGLE VALUE IN KON, PROCOM, LENGTH OR STATUS

SUBROUTINE SETLIM (ITIME,IPAGE,IFILE,IQ)
  ZEROES STATUS COMMON AND SETS THE LIMITS

SUBROUTINE SETKSD (M,L,NA,NB,NC,ND,NE,NF,NG,NH)
  ENTERS DATA INTO K OF COMMON KON
```

```
SUBROUTINE SETKSX (J,L,IS,ID,IL)
  COPIES SELECTED DATA FROM AN ARRAY J INTO KON. IT IS
  USED ONLY BY SETKON.

SUBROUTINE SETPSD (M,L,NA,NB,NC,ND,NE,NF,NG,NH)
  ENTERS DATA INTO PROCOM OR LENGTH

SUBROUTINE SETSYM (IB,IS,NL,NM,ITAB,IALT)
  FILLS KSYM WITH PLOTTING SYMBOLS

SUBROUTINE PRORDY (ML,LDL,LPL,LPPL,JA,JB,JD,JP)
  PERFORMS THE INITIALISATIONS BEFORE A PLOT. IT CAN ALSO
  PERFORM FINAL WRAP UP PROCEDURES. THIS ROUTINE IS CALLED
  FROM THE MAIN ROUTINE (EXPLOT).

SUBROUTINE PROSEQ (ML)
  GENERATES THE TYPE ML OF THE NEXT LINE TO BE PROCESSED, K
  BEING THE SIMILAR INTERNAL VARIABLE CONTROLLING THE SEQUENCE.
  DUMMY LINES ARE PRODUCED AT THE END OF EACH BLOCK OF OUTPUT,
  OF A DIFFERENT TYPE AT THE END OF THE ENTIRE PLOT. THIS
  ROUTINE IS CALLED FROM THE MAIN ROUTINE (EXPLOT).

SUBROUTINE PROLN (ML,LDL,LOL,LPL,JA,JB,JD,JO,JP,JQ)
  PROCESSES A LINE OF OUTPUT, REAL OR DUMMY, ACCORDING TO THE
  TYPE SPECIFIED BY ML. ARRAYS WITH ADJUSTABLE DIMENSIONS AND
  THEIR SIZES ARE GIVEN AS PARAMETERS. OTHER DATA IS PASSED IN
  COMMON. THIS ROUTINE IS CALLED FROM THE MAIN ROUTINE (EXPLOT).

SUBROUTINE EXPLOT
  CAUSES PLOTTING ACCORDING TO THE PRESET COMMON VARIABLES.

SUBROUTINE NEWLN (MSS,MA,MX,J,L)
  SUPPLIES A LINE OF DATA.
  IF MSS POSITIVE IT COMES FROM A FILE MOD(MSS,100) WITH ZERO
  DEFAULTING TO TAPE7.
  IF MSS NEGATIVE THE DATA IS INTERNALLY GENERATED DUMMY.
  IF THE THIRD DIGIT FROM THE RIGHT OF MSS IS NOT ZERO THE
  FILE IS NOT REWOUND DURING INITIALISATION.
  IF THE FOURTH DIGIT FROM THE RIGHT OF MSS IS NOT ZERO THE
  DATA IS SCALED BEFORE CONVERSION TO INTEGER BY 10**D,
  OTHERWISE BY 1000.
  THIS ROUTINE IS CALLED FROM LNDATA

SUBROUTINE BORDER (NB,NLAP,KSYM,MB,IOL,IIL,ILAP)
  RETURNS BORDER AND OVERLAP WIDTHS (SIDE), AND A SIDE BORDER
  SYMBOL DEPENDING ON LINE, SET BY MB. THIS ROUTINE IS CALLED
  FROM A NUMBER OF OTHER ROUTINES.

SUBROUTINE ERROR
  PRINTS, TO UNIT 6, THE ERROR TERMINATION NUMBER ON A NEW
  PAGE, FOLLOWED BY THE STATUS VARIABLES, AND STOPS. IT CAN
  BE CALLED FROM ANY ROUTINE WHEN AN ERROR IS DETECTED, IF
  IEQ=1.
```

```
SUBROUTINE BLKEND (ISEC,NSEC,IBLK,NBLK,KIDA,KIDB,MF,ML,MO,JP,LPL)
  PRODUCES A LABEL LINE AFTER A BLOCK OF DATA HAS BEEN PLOTTED.
  THE FIRST SIX PARAMETERS PROVIDE THE TEXT, MF GIVES THE
  TYPE OF LABEL AND DESTINATION (EITHER OF TWO PRINT FILES OR
  THE MERGE FILE) ALONG WITH MO INDICATING MICROFILM OR
  PRINTER. FOR MICROFILM A CALL IS MADE TO MFEND. ML INDICATES
  END OF BLOCK OR END OF DATA AND ALONG WITH THE FIRST FOUR
  PARAMETERS CONTROLS THE PRODUCTION OF A LABEL. THIS
  ROUTINE IS CALLED FROM PROLN.

SUBROUTINE LNSPEC (IIL,IOL,NOL,NDL,MO)
  RETURNS FOR OUTPUT LINE NUMBER IOL (OF NOL OUTPUT LINES) A
  POSITION IIL AMONG THE NDL INPUT DATA LINES, EXPRESSED IN
  UNITS OF 1/128. IT IS CALCULATED ACCORDING TO THE METHOD
  SPECIFIED BY MO, AND THE SIGN OF IIL INDICATES THE WAY THE
  OUTPUT LINE SHOULD BE GENERATED. POSITIVE IS INTERPOLATED,
  NEGATIVE IS GROUPED OR INTEGRATED (INTEGRATED NOT PRESENTLY
  IMPLEMENTED). THIS ROUTINE IS CALLED FROM PRORDY AND PROLN.

SUBROUTINE LNDATA (JA,JB,JD,LDL,IIL,MO,MS)
  PLACES IN JD A LINE OF DATA DERIVED ACCORDING TO THE
  INTERPOLATED POSITION IIL FROM THE INPUT DATA ACCESSED BY
  NEWLN. FOR IIL=0 THE SOURCE IS INITIALISED. JA, JB ARE USED
  FOR LOCAL STORAGE OF LINES OF INPUT DATA. MS ALLOWS DATA
  SOURCE TO BE SELECTED, MO CAN SPECIFY HIGHER ORDER
  INTERPOLATION (NOT IMPLEMENTED). THIS ROUTINE IS CALLED
  FROM PROLN.

SUBROUTINE LNSCAL (JO,LOL,JD,LDL,MO)
  TAKES DATA IN JD AND ADJUSTS IT IN LENGTH TO FIT JO
  USING THE METHOD INDICATED BY MO. IF LOL IS NOT
  CORRECTLY RELATED TO LDL IT IS TREATED AS AN ERROR.
  ACCEPTABLE ARE NOMINAL INTEGRAL RATIOS EITHER WAY, 3/2
  AND 2/3. THIS ROUTINE IS CALLED FROM PROLN.

SUBROUTINE LNBORD (JP,LPL,LOL,KSEC,LSEC,MSEC,NSEC,MB,MF,ML,KPL)
  PRODUCES ONE OR MORE OUTPUT BORDER LINES OF TYPE SPECIFIED
  BY ML AND ACCORDING TO THE SECTIONING SPECIFICATION KSEC...
  IT WRITES THEM TO THE SECTION FILES, OR LEAVES ONE IN JP.
  THIS ROUTINE IS CALLED FROM PROLN.

SUBROUTINE LNOUT (JP,LPL,JO,LOL,KSEC,LSEC,MSEC,NSEC,MB,MF,IOL,IIL,ILAP,KPL)
  PRODUCES ONE OR MORE OUTPUT LINES ACCORDING TO THE
  SECTIONING SPECIFICATION KSEC...(COMPLETE WITH OVERLAP
  AND BORDER IF REQUIRED) FROM JO, AND WRITES THEM TO
  THE SECTION FILES, OR LEAVES ONE IN JP. THIS ROUTINE
  IS CALLED FROM PROLN.

SUBROUTINE LNPLOT (JP,JQ,LPL,MF,MO,MP)
  OUTPUTS A LINE OF DATA FROM JP IN MODE SPECIFIED BY MO.
  JQ IS USED BT THE CONTOURING ROUTINES FOR STORING THE
  PREVIOUS LINE OF DATA. THIS ROUTINE IS CALLED FROM PROLN.
```

```
SUBROUTINE LPRTZ (JP,LPL,MF)
  CONVERTS THOSE VALUES IN JP WHICH REPRESENT DATA INTO VALUES
  FROM KONSYM, EACH REPRESENTING ONE OR MORE PRINTING SYMBOLS
  TO BE SUPERIMPOSED. JP IS THEN PASSED TO LPRT, WITH THE
  NUMBER OF OVERPRINTED OUTPUT LINES SET TO SUIT KONSYM, OR 1
  IF NO VALUES REPRESENTED DATA. VALUES OUTSIDE THE RANGE OF
  DATA ARE PASSED UNALTERED. THIS ROUTINE IS CALLED FROM
  LNPLOT.

SUBROUTINE LPRT (JP,LPL,MF,NP)
  OUTPUTS THE DATA IN JP TO A PRINT FILE SELECTED BY MF AS NP
  OVERPRINTED LINES, EACH LINE TAKING THE LEFTMOST REMAINING
  CHARACTER FROM EACH ELEMENT OF JP. IN MERGE MODE EACH OUTPUT
  LINE IS PASSED INSTEAD TO LMERGE FOR INCORPORATION IN THE
  MERGE FILE. THIS ROUTINE IS CALLED FROM LPRTZ AND LPRTC.

SUBROUTINE LMERGE (JP,LPL)
  READS A LINE FROM (91) AND COPIES TO (92) WITH JP APPENDED.
  FILES ARE READ A LINE AT A TIME, AND THE LAST VALUE GIVES
  THE NUMBER OF POSITIONS ALREADY USED. IF A LINE IS EMPTY
  (NUMBER=1) THE PRINTER CONTROL VALUE IS COPIED INTO THE
  FIRST PLACE, OTHERWISE IT IS DISCARDED. SOURCE FILE IS
  REWOUND WHEN IT REACHES THE END, AN EMPTY BLANK LINE
  BEING SUBSTITUTED. THIS ROUTINE IS CALLED FROM LPRT AND PROLN.

SUBROUTINE LDMP (JP,LPL,MF)
  OUTPUTS THE DATA IN JP TO A PRINT FILE SELECTED BY MF, AS
  INTEGERS. VALUES OUTSIDE THE DATA RANGE ARE CONVERTED TO
  SMALL INTEGERS. THE FIRST VALUE IS TRANSMITTED IN A1
  FORMAT FOR PRINTER CONTROL. THIS ROUTINE IS CALLED FROM
  LNPLOT.

SUBROUTINE MFZ (JP,LPL,MP)
  PLOTS JP ON MICROFILM. SYMBOLS ARE RECOGNISED AND DRAWN. DATA
  IS REPRESENTED BY SHAPES OF VARYING SIZE, THERE BEING A NUMBER
  OF SHAPES AND PATTERNS AVAILABLE. UP TO THREE RANGES OF DATA,
  PLOTTED AT DIFFERENT INTENSITY, ARE POSSIBLE TO EXTEND THE
  DYNAMIC RANGE. THIS ROUTINE IS CALLED FROM LNPLOT AND MFGS.

SUBROUTINE MFC (JP,JQ,LPL,MP)
  PLOTS JP ON MICROFILM AS CONTOURS (IN COMPARISON WITH PREVIOUS
  LINE JQ), CALLING MFCL OR MFCSP FOR THE ACTUAL CONTOURING.
  SYMBOLS ARE RECOGNISED AND DRAWN. CURRENT DATA IS COPIED
  INTO JQ. IT REQUIRES THAT ANY SEQUENCE OF LINES OF DATA BE
  PRECEDED BY A PURELY SYMBOLIC LINE. THIS ROUTINE IS CALLED
  FROM LNPLOT AND MFGS.

SUBROUTINE MFCL (IP,IQ,KX,KY,IX,IY,N)
  CONTOURS ON MICROFILM ONE CELL OF DATA, DIMENSIONS IX,IY,
  LOWER RIGHT CORNER AT KX,KY. LOWER RIGHT DATA VALUE IS IP,
  UPPER RIGHT IQ. THE LEFT VALUES ARE ASSUMED SUPPLIED BT THE
  PREVIOUS CALL. THE COUNT N OF SUCCESSIVE DATA VALUES IN THE
  CURRENT (LOWER) LINE ENABLES THE FIRST CALL, WITH NO
  PRECEDING DATA, TO BE RECOGNISED. THIS ROUTINE IS CALLED
  FROM MFC.
```

SUBROUTINE MFPAGE (MF,MO,MP,LPL)
SETS UP A NEW FRAME ON MICROFILM. IF THE FIRST FRAME IT MAKES THE CALL TO START, AND IF THE LAST HAS BEEN REACHED IT TAKES NO ACTION. IF IT IS CALLED AFTER THE PLOT HAS BEEN CLOSED (MARKED BY A NEGATIVE FRAME COUNT) IT IS TREATED AS AN ERROR. THE ORIGIN POSITION IS SET IN IES. IF REQUIRED BY MF A CALL TO MFGS SETS UP A BORDER OR A GREY SCALE. THIS ROUTINE IS CALLED FROM PRORDY, MFZ, AND MFC.

SUBROUTINE MFGS (MO,MP)
GENERATES A STEPPED OR CONTINUOUS SCALE OF VALUES AND CAUSES IT TO BE OUTPUT TO MICROFILM, WITH OR WITHOUT NUMERIC LABELS. THE TYPE IS GIVEN BY MP. IT CAN ALSO DRAW A PARTIAL BORDER ROUND THE ENDS OF THE FRAME. THIS ROUTINE IS CALLED FROM MFPAGE.

SUBROUTINE MFEND (ISEC,NSEC,IBLK,NBLK,KIDA,KIDB,K)
PRODUCES A LABEL ON MICROFILM EITHER BELOW THE PLOT OR IN THE LOWER RIGHT CORNER ACCORDING TO THE PARAMETER K. THE OTHER PARAMETERS PROVIDE THE TEXT. THIS ROUTINE IS CALLED FROM BLKEND.

SUBROUTINE MFSYM (KX,KY,IX,IY,IA,IT,M,N)
PLOTS ON MICROFILM CHARACTER STRINGS, NUMBERS, SYMBOLS, LINES AND POINTS ACCORDING TO MODE M AND VALUE N, AT INTENSITY IT, POSITION ANGLE (FOR TEXT) IA, AND POSITION KX, KY. CHARACTER SIZE IS SPECIFIED BY IX, IY. THIS ROUTINE IS USED BY MOST OF THE MICROFILM OUTPUT ROUTINES.

SUBROUTINE MBX (KL,KR,KB,KT,IT)
PLOTS A RECTANGLE ON MICROFILM, INTENSITY IT, DIAGONAL CORNERS KL,KB KR,KT.

```
      SUBROUTINE PLOTZ (LL,NL,KBASE,KRANGE,KIDA,KIDB,MS,NGS)
C PLOTS FROM FILE MS ONTO MICROFILM.
      CALL SETZ(LL,NL,KBASE,KRANGE,KIDA,KIDB,MS,NGS)
      CALL EXPLOT
      RETURN
      END
C
      SUBROUTINE PLOTC (LL,NL,KBASE,KRANGE,KIDA,KIDB,MS,NGS,KLOW)
C PLOTS LOG CONTOURS FROM FILE MS ONTO MICROFILM.
      CALL SETC(LL,NL,KBASE,KRANGE,KIDA,KIDB,MS,NGS,KLOW)
      CALL SETBOX
      CALL SETLOG
      CALL EXPLOT
      RETURN
      END
C
      SUBROUTINE SETZ (LL,NL,KBASE,KRANGE,KIDA,KIDB,MS,NGS)
C PREPARES TO PLOT FROM MS (0 DEFAULTING TO TAPE7) ON MICROFILM.
      NS=MAXO(LL,NL)
      MOP=700+((NS+20)/15+1)
      IF(NGS.GT.0) MOP=MOP-600
      CALL SETCOM(0,0,0,KBASE,KRANGE,KIDA,KIDB)
      CALL SETBOR(428,9,9,1,1)
      CALL SETDIM(LL,LL,LL+20,LL+20,0,NL,NL,NL+20)
      CALL SETMOP(4,500,MOP,MS,KIDA,KIDB)
      RETURN
      END
C
      SUBROUTINE SETC (LL,NL,KBASE,KRANGE,KIDA,KIDB,MS,NGS,KLOW)
C PREPARES TO PLOT FROM MS (0 DEFAULTING TO TAPE7) AS CONTOURS
C ON MICROFILM.
      NS=MAXO(LL,NL)
      MOP=900+((NS+20)/15+1)
      IF(NGS.GT.0) MOP=MOP-800
      CALL SETCOM(0,0,0,KBASE,KRANGE,KIDA,KIDB)
      CALL SETBOR(428,9,9,1,1)
      CALL SETDIM(LL,LL,LL+20,LL+20,0,NL,NL,NL+20)
      CALL SETMOP(4,600,MOP,MS,KIDA,KIDB)
      CALL SETVAL(24,KLOW)
      RETURN
      END
C
      SUBROUTINE SETBOR (MB,NB,NBBLK,NLAP,NLPBLK)
C SETS THE BORDER MODE AND PARAMETERS
      CALL SETPSD(7,5,MB,NB,NBBLK,NLAP,NLPBLK,0,0,0)
      RETURN
      END
C
      SUBROUTINE SETBOX
C SETS BOX BORDER
      CALL SETBOR(286,1,1,1,1)
      RETURN
      END
C
      SUBROUTINE SETDIM (LDL,LOL,LPL,LPPL,KPL,NDL,NOL,NPL)
C SETS THE LINE LENGTHS AND THE NUMBERS OF LINES
      CALL SETPSD(1,-4,LDL,LOL,LPL,LPPL,0,0,0,0)
      CALL SETPSD(12,4,NDL,NOL,NPL,KPL,0,0,0,0)
      RETURN
      END
C
      SUBROUTINE SETLAB (KIDA,KIDB)
C SETS THE IDENTITY NUMBERS
      CALL SETPSD(1,2,KIDA,KIDB,0,0,0,0,0,0)
      RETURN
      END
C
      SUBROUTINE SETLOG
C SETS LOG SWITCHES.
      CALL SETVAL(9,1)
      CALL SETVAL(17,1)
      CALL SETVAL(26,1)
      CALL SETVAL(44,1)
      RETURN
      END
C
      SUBROUTINE SETMOP (MF,MO,MP,MS,KIDA,KIDB)
C SETS THE MODES (EXCEPT MB) AND IDENTITY NUMBERS
      CALL SETPSD(1,6,KIDA,KIDB,MF,MO,MP,MS,0,0)
      RETURN
      END
C
      SUBROUTINE SETSEC (LBLK,MBLK,NBLK,LSEC,MSEC,NSEC,KBLK,KSEC)
C SETS THE SECTIONING PARAMETERS
      CALL SETPSD(16,8,LBLK,MBLK,NBLK,LSEC,MSEC,NSEC,KBLK,KSEC)
      RETURN
      END
C
      SUBROUTINE SETPRZ
C SETS PROCOM TO ZERO
      CALL SETPSD(0,0,0,0,0,0,0,0,0,0)
      RETURN
      END
C
      SUBROUTINE SETCOM (MKON,MBOP,MSDIM,KBASE,KRANGE,KIDA,KIDB)
C SETS ALL THE COMMON ARRAYS
      CALL SETKON(MKON,KBASE,KRANGE)
      CALL SETPRO(MBOP,MSDIM,KIDA,KIDB)
      RETURN
      END
C
      SUBROUTINE SETKON (M,KB,KR)
C SETS KON (K AND KSYM) WITH A SET OF VALUES.
      INTEGER J(66)
      DATA LJ,(J(I),I=1,66)/66,1000,0,1,0,
     + 0,   0,   1,   0,1000,   0,   8,   0,   0,   2,
     + 0,   0,   0, 500,   0,   5,   0,   0,   4,   1,
     + 2,   1,  10,  20,  18,  24,  20,  30,   0,   0,
     + 0,1000,   0,   1,   0,   0,   0,   4,   0,  82,
     + 20,2048,   0, 215,   0,  10,-100000,1024,0,   0,
     + 0,  20,-100000,1024,0,   0,   0,  30,100000,1024,   0,   0/
C***ZERO,ENTER BASE AND RANGE, ENTER MODE IDENT ACCORDING TO
C***MODE M
      IF(M.GE.-1) CALL SETKSD(0,0,0,0,0,0,0,0,0,0)
      IF(M.GE.-2) CALL SETKSD(2,2,KB,KR,0,0,0,0,0,0)
      IF(M.GE.1) CALL SETKSD(1,1,M,0,0,0,0,0,0,0)
C***SET K, KSYM
      IF(M.GE.0) CALL SETKSX(J,LJ,1,6,66)
      IF(M.GE.0) CALL SETSYM(4,1,1,32,0,0)
      RETURN
      END
C
      SUBROUTINE SETPRO (MBOP,MSDIM,KIDA,KIDB)
C SETS PROCOM WITH A SET OF VALUES
C THIS VERSION IS A DUMMY
      IF(MBOP.LT.0.OR.MSDIM.LT.0) THEN
        CALL SETLAB(KIDA,KIDB)
        RETURN
      ENDIF
      CALL SETPRZ
      CALL SETLIM(50,45,98000,2)
      CALL SETBOR(612,3,3,3,3)
      CALL SETMOP(0,0,200,0,KIDA,KIDB)
      CALL SETSEC(500,0,1,500,0,1,0,0)
      CALL SETDIM(60,60,67,70,0,60,48,80)
      RETURN
      END
C
      SUBROUTINE SETVAL (I,N)
C SETS A SINGLE VALUE IN KON, PROCOM, LENGTH OR STATUS
      COMMON/KON/K(128),KLEV(128),KSYM(128),KTAB(2048)
      COMMON/PROCOM/IP(30)
      COMMON/LENGTH/L(4)
      COMMON/STATUS/IEQ,IER,IES(32)
      IF(I.EQ.0) IEQ=N
      IF(I.GT.128.AND.I.LE.160) IES(I-128)=N
      IF(I.GT.0.AND.I.LE.128) K(I)=N
      IF(I.LT.0.AND.I.GE.-30) IP(-I)=N
      IF(I.LT.-30.AND.I.GE.-34) L(-I-30)=N
      RETURN
      END
C
      SUBROUTINE SETLIM (ITIME,IPAGE,IFILE,IQ)
C ZEROES STATUS COMMON AND SETS THE LIMITS
      COMMON/STATUS/IEQ,IER,IES(32)
      DATA JFIRST/1/
      IEQ=IQ
      IER=0
      IF(JFIRST.GT.0) THEN
        JFIRST=0
        DO 100 I=1,32
        100 IES(I)=0
      ENDIF
      IES(4)=IFIX(SECOND(X))
      IES(5)=IES(4)+ITIME
      IES(6)=IPAGE
      IES(7)=IFILE
      IES(30)=999999999
      IES(31)=1999
      IES(32)=20
      RETURN
      END
C
```

```
      SUBROUTINE SETKSD (M,L,NA,NB,NC,ND,NE,NF,NG,NH)
C ENTERS DATA INTO K OF COMMON KON
      COMMON/KON/K(128),KLEV(128),KSYM(128),KTAB(2048)
      COMMON/STATUS/IEQ,IER,IES(32)
      INTEGER N(8)
      N(1)=NA
      N(2)=NB
      N(3)=NC
      N(4)=ND
      N(5)=NE
      N(6)=NF
      N(7)=NG
      N(8)=NH
      IF(L.LE.0.OR.M.LE.0) THEN
        DO 100 I=1,128
        100 K(I)=0
      ELSEIF((L+M).GT.129) THEN
        IER=8001
        IF(IEQ.EQ.1) CALL ERROR
      ELSE
        J=M
        DO 110 I=1,MINO(L,8)
          K(J)=N(I)
          J=J+1
        110 CONTINUE
      ENDIF
      RETURN
      END
C
      SUBROUTINE SETKSX (J,L,IS,ID,IL)
C COPIES SELECTED DATA FROM AN ARRAY J INTO KON. IT IS
C USED ONLY BY SETKON.
      INTEGER J(L)
      COMMON/KON/K(128),KLEV(128),KSYM(128),KTAB(2048)
      M=IS
      N=ID
      DO 100 I=1,IL
        K(N)=J(M)
        M=M+1
        N=N+1
      100 CONTINUE
      RETURN
      END
C
      SUBROUTINE SETPSD (M,L,NA,NB,NC,ND,NE,NF,NG,NH)
C ENTERS DATA INTO PROCOM OR LENGTH
      INTEGER IDX(23),N(8)
      COMMON/PROCOM/IP(30)
      COMMON/LENGTH/LEN(4)
      DATA (IDX(I),I=1,23)/29,30,22,23,24,25,21,13,6,14,7,18,
     +                     19,20,28,3,4,5,10,11,12,2,9/
      N(1)=NA
      N(2)=NB
      N(3)=NC
      N(4)=ND
      N(5)=NE
      N(6)=NF
      N(7)=NG
      N(8)=NH
      IF(L.EQ.0.OR.M.EQ.0) THEN
C  ***ZERO PROCOM
        DO 100 I=1,30
        100 IP(I)=0
      ELSEIF(L.GT.0.AND.M.GT.0) THEN
C  ***INDEXED ENTRY INTO PROCOM
        K=M
        DO 110 I=1,MINO(L,8)
          IF(K.LE.23) IP(IDX(K))=N(I)
          K=K+1
        110 CONTINUE
      ELSEIF(L.GT.0.AND.M.LT.0) THEN
C  ***ENTRY INTO PROCOM IN ORDER
        K=-M
        DO 120 I=1,MINO(L,8)
          IF(K.LE.30) IP(K)=N(I)
          K=K+1
        120 CONTINUE
      ELSEIF(L.LT.0.AND.M.GT.0) THEN
C  ***FILL LENGTH
        DO 130 I=1,4
        130 LEN(I)=N(I)
      ELSEIF(L.LT.0.AND.M.LT.0) THEN
C  ***ENTRY INTO LENGTH AS SPECIFIED
        K=-M
        DO 140 I=1,MINO(-L,8)
          IF(K.LE.4) LEN(K)=N(I)
          K=K+1
        140 CONTINUE
      ENDIF
      RETURN
      END
C
      SUBROUTINE SETSYM (IB,IS,NL,NM,ITAB,IALT)
C FILLS KSYM WITH PLOTTING SYMBOLS
      COMMON/KON/K(128),KLEV(128),KSYM(128),KTAB(2048)
      INTEGER KA(32)
      DATA KBLANK/1H /
      DATA (KA(I),I=1,32)/1H0,1H.,1H1,1H.,1H2,1H-,1H3,1H+,
     +  1H4,1H.,1H5,1H.,1H6,1H-,1H7,1H+,1H8,1H.,1H9,1H.,
     +  1HA,1H-,1HB,1H+,1HC,1H.,1HD,1H.,1HE,1H-,1HF,1H+/
      NB=IB
      IF(NB.EQ.0) NB=1
      MX=32
      DO 100 I=1,128
        N=(((I-1)/NB)*IS)+NL
        N=MAXO(1,MINO(N,NM))
        N=1+MOD((N-1),MX)
        KSYM(I)=KA(N)
        IF(ITAB.EQ.-1) KSYM(I)=KA(MAXO(1,MINO(NL,32)))
        IF(MOD(N,2).EQ.0.AND.IALT.EQ.1) KSYM(I)=KBLANK
      100 CONTINUE
      RETURN
      END
C
```

```
      SUBROUTINE PRORDY (ML,LDL,LPL,LPPL,JA,JB,JD,JP)
C
C PERFORMS THE INITIALISATIONS BEFORE A PLOT. IT CAN ALSO
C PERFORM FINAL WRAP UP PROCEDURES. THIS ROUTINE IS CALLED
C FROM THE MAIN ROUTINE (EXPLOT).
C
      INTEGER JA(LDL),JB(LDL),JD(LDL),JP(LPPL)
      COMMON JM(150)
      COMMON/STATUS/IEQ,IER,IES(32)
      COMMON/PROCOM/IBLK,KBLK,LBLK,MBLK,NBLK,NBBLK,NLPBLK,
     +              ISEC,KSEC,LSEC,MSEC,NSEC,NB    ,NLAP  ,
     +              IDL,IOL,IPL,NDL,NOL,NPL,
     +              MB,MF,MO,MP,MS,IIL,ILAP,KPL,KIDA,KIDB
      DATA KBLANK/1H /
C
C***HEADER, OUTPUT CODES. PRINTER, MICROFILM ACTIVE FLAGS
      KH=MOD((MF/10),10)
      KO=MOD((MF/100),10)
      M=MOD((MO/100),10)
      KP=0
      IF(M.LE.4.AND.KO.NE.3) KP=1
      KM=0
      IF(M.GE.5.AND.KO.NE.3) KM=1
C***PRINTER OUTPUT FILE
      IU=10
      IF(MOD((MF/2000),2).EQ.1) IU=9
C
C***INITIALISE SOURCE AND LNDATA IF THEY WILL BE USED
      IF(KH.NE.9.AND.KO.LE.7) THEN
        CALL LNDATA(JA,JB,JD,LDL,0,MO,MS)
C  ***HYPOTHETICAL PRECEDING OUTPUT LINE
        IOL=KBLK+MAXO(0,(MBLK-1))*LBLK
        IOL=MINO(IOL,(NOL-1))
        IF(IOL.NE.0) THEN
C         ***ADVANCE TO CORRECT START POSITION
          CALL LNSPEC(IIL,IOL,NOL,NDL,MO)
          IF(IIL.LT.0) IIL=-IIL-64
          IF(IIL.NE.0) CALL LNDATA(JA,JB,JD,LDL,IIL,MO,MS)
        ENDIF
      ENDIF
C
C***REWIND ANY SECTION FILES THAT WILL BE USED
      IF(KH.LE.8.AND.KO.GE.3.AND.KO.LE.8) THEN
        M=1
        N=NSEC
        IF(MSEC.GT.0) M=MSEC
        IF(MSEC.GT.0) N=MSEC
        DO 100 I=M,N
          IR=I+10
          REWIND IR
  100   CONTINUE
      ENDIF
C
C***MERGE FILES USED (MERGE MODE AND TO PRINTER)
      IF(KP.EQ.1.AND.KO.EQ.1) THEN
C  ***NEW PLOT. INITIALISE SOURCE FILE WITH 30 EMPTY LINES
        IF(KH.LE.6) THEN
          REWIND 91
          DO 200 I=1,150
          200 JM(I)=KBLANK
          JM(150)=1
          DO 210 I=1,30
          210 WRITE(91) JM
          REWIND 91
          REWIND 92
C  ***ADD BLANK LINES TO MERGE FILE
        ELSEIF(KH.EQ.7) THEN
          DO 220 I=1,LPL
          220 JP(I)=KBLANK
          DO 230 I=1,5
          230 CALL LMERGE(JP,LPL)
C  ***RESET. COPY DESTINATION MERGE FILE BACK TO SOURCE
        ELSEIF(KH.EQ.8) THEN
          REWIND 91
          REWIND 92
          240 CONTINUE
            READ(92,END=250) JM
            WRITE(91) JM
          GO TO 240
          250 CONTINUE
          REWIND 91
          REWIND 92
C  ***WRAP. OUTPUT MERGED FILE TO PRINTER
        ELSEIF(KH.EQ.9) THEN
          REWIND 92
          L=MINO(149,LPPL)
          260 CONTINUE
C           ***RETURN IF OUT OF TIME
            IF(SECOND(X).GE.FLOAT(IES(5))) RETURN
            READ(92,END=270) JM
C           ***SET DUMMY MODE FOR CORRECT DESTINATION
            M=MOD((MF/2000),2)*2000
            CALL LPRT(JM,L,M,1)
          GO TO 260
          270 CONTINUE
        ENDIF
      ENDIF
C
C***FINAL WRAP (NOT MERGE)
      IF(KH.EQ.9.AND.KO.NE.1) THEN
C  ***IF MICROFILM USED, CLOSE PLOT AND SET FRAME COUNT -VE
        IF(IES(11).GT.0) CALL ENPLOT
        IF(IES(11).GT.0) IES(11)=-IES(11)
C  ***IF PRINTER OUTPUT, WRITE A NEW PAGE
        IF(IES(10).GT.0) WRITE(10,1300)
        IF(IES(9).GT.0) WRITE(9,1300)
        1300 FORMAT(1H1)
      ENDIF
C
C***MICROFILM USED
      IF(KM.EQ.1) THEN
C  ***SEPARATE BASIC MP AND GREY SCALE PARAMETER
        M=1000*(MP/1000)+MOD(MP,200)
        N=MOD((MP/200),5)
C  ***ADD GREY SCALE IF REQUIRED BY CHANGING M WITH KH,MP
        IF(KH.EQ.0.OR.KH.EQ.6) THEN
          M=MP
        ELSEIF(KH.EQ.2.OR.KH.EQ.4) THEN
          M=M+600
          IF(N.NE.0) M=MP
        ELSEIF(KH.EQ.5) THEN
          M=M+800
          IF(N.NE.0) M=MP
        ENDIF
C  ***NEW FRAME IF REQUIRED
        IF(KH.LE.6) CALL MFPAGE(MF,MO,M,LPL)
C  ***HEADER IF REQUIRED
        IF(KH.GT.0.AND.KH.LE.6) CALL MFHD(KH)
C  ***MOVE ORIGIN IF REQUIRED
        IF(KH.EQ.7) THEN
C         ***CONTINUE DOWN FRAME AFTER A GAP
          IES(16)=MAXO(0,(IES(16)-440))
        ELSEIF(KH.EQ.8.AND.KO.EQ.1) THEN
C         ***RESET (MERGE). RESET ORIGIN TO TOP, RIGHT OF LAST PLOT
          IES(13)=IES(15)+1100
          IES(16)=IES(14)
        ELSEIF(KH.EQ.8.AND.KO.EQ.0) THEN
C         ***RESET (NON MERGE) TO SUPERIMPOSE ON PREVIOUS PLOT
          IES(16)=IES(14)
        ENDIF
      ENDIF
C
C***PRINTER USED
      IF(KP.EQ.1) THEN
        IF(KH.LE.6) THEN
C         ***BEGIN ON NEW PAGE
          WRITE(IU,1400)
          1400 FORMAT(1H1)
          IES(IU)=IES(IU)+1
C         ***SET OR CLEAR AUTO PAGE EJECT IF PERMITTED
          IF(MOD((MP/1000),2).EQ.1) THEN
            IF(MOD((MF/1000),2).EQ.1) THEN
C             ***CLEAR AUTO
              WRITE(IU,1410)
              1410 FORMAT(1HQ)
            ELSE
C             ***SET AUTO
              WRITE(IU,1420)
              1420 FORMAT(1HR)
            ENDIF
            IES(IU)=IES(IU)+1
          ENDIF
C         ***SET PITCH IF PERMITTED
          IF(MOD((MP/2000),2).EQ.1) THEN
            IF(MOD((MP/100),10).LE.1) THEN
C             ***CLEAR 8 LINES/INCH
              WRITE(IU,1430)
              1430 FORMAT(1HS)
            ELSE
C             ***SET 8 LINES/INCH
              WRITE(IU,1440)
              1440 FORMAT(1HT)
            ENDIF
            IES(IU)=IES(IU)+1
          ENDIF
C         ***PRINT HEADER IF REQUIRED
          IF(KH.GT.0) CALL PRTHD(KH)
        ELSEIF(KH.EQ.7.AND.KO.NE.1) THEN
C         ***CONTINUE WITH BLANK LINES FOR SPACING
          WRITE(IU,1450)
          1450 FORMAT(1H ,////)
          IES(IU)=IES(IU)+4
        ENDIF
      ENDIF
C
C***SET 'NO NEW PAGE' FLAG
      IES(3)=0
      IF(KH.EQ.0.OR.KH.GE.6) IES(3)=1
C
C***INITIALISE LOOPING PARAMETERS FOR PROSEQ
      IOL=0
      ILAP=0
      ML=8
      IBLK=MAXO(1,MBLK)-1
C
      RETURN
      END
C*** OF PRORDY
```

```
      SUBROUTINE PROSEQ (ML)
C
C GENERATES THE TYPE ML OF THE NEXT LINE TO BE PROCESSED, K
C BEING THE SIMILAR INTERNAL VARIABLE CONTROLLING THE SEQUENCE.
C DUMMY LINES ARE PRODUCED AT THE END OF EACH BLOCK OF OUTPUT,
C OF A DIFFERENT TYPE AT THE END OF THE ENTIRE PLOT. THIS
C ROUTINE IS CALLED FROM THE MAIN ROUTINE (EXPLOT).
C
      COMMON/PROCOM/IBLK,KBLK,LBLK,MBLK,NBLK,NBBLK,NLPBLK,
     +              ISEC,KSEC,LSEC,MSEC,NSEC,NB    ,NLAP  ,
     +              IDL,IOL,IPL,NDL,NOL,NPL,
     +              MB,MF,MO,MP,MS,IIL,ILAP,KPL,KIDA,KIDB
C
C***INCREMENT COUNTERS FOR NEXT LINE
      IOL=IOL+1
      ILAP=ILAP+1
C
C***LAST LINE A DUMMY. SET TO FIRST BORDER LINE OF NEXT BLOCK
      IF(ML.GE.8) THEN
        K=1
        ILAP=1
        IBLK=IBLK+1
C  ***FIRST AND LAST OUTPUT DATA LINES
        IOLA=MINO(NOL,((IBLK-1)*LBLK+KBLK+1))
        IOLD=MINO(NOL,(IOLA+LBLK+NLPBLK-1))
C  ***FIRST AND LAST OUTPUT LINES INSIDE OVERLAPS
        IOLB=MINO(NOL,(IOLA+NLPBLK))
        IOLC=MINO(NOL,(IOLD-NLPBLK))
C  ***SET TYPE AS FULL OR BLANK BORDER LINE
        ML=3
        IF(NBBLK.LE.0) ML=4
      ENDIF
C***IF A LINE OF TOP BORDER AFTER FIRST, ALTER TO BORDER LINE
C***WITH SIDE SYMBOLS ONLY
      IF(K.EQ.1.AND.ILAP.GT.1) THEN
        K=2
        ML=0
      ENDIF
C***IF A LINE OF TOP BORDER AFTER FIRST AND PAST WIDTH OF TOP
C***BORDER, ALTER TO FIRST LINE OF TOP OVERLAP
      IF(K.EQ.2.AND.ILAP.GT.NBBLK) THEN
        K=3
        IOL=IOLA
        ILAP=1
        ML=5
C  ***IF FIRST BLOCK, LINE IS NOT TO BE READ FROM SCRATCH FILE
        IF(IBLK.EQ.MAXO(1,MBLK)) ML=7
      ENDIF
C***IF A LINE OF TOP OVERLAP AND PAST THE WIDTH OF TOP OVERLAP,
C***ALTER TO NORMAL DATA LINE
      IF(K.EQ.3.AND.IOL.GE.IOLB) THEN
        K=4
        ML=7
      ENDIF
C***IF NORMAL DATA LINE AND INTO BOTTOM OVERLAP, ALTER
C***TO BOTTOM OVERLAP
      IF(K.EQ.4.AND.IOL.GT.IOLC) THEN
        K=5
        ILAP=1
        ML=6
C  ***IF LAST BLOCK, LINE IS NOT TO BE COPIED TO SCRATCH FILE
        IF(IBLK.GE.NBLK.OR.(IBLK.GE.MBLK.AND.MBLK.GT.0)) ML=7
      ENDIF
C***IF A LINE OF BOTTOM OVERLAP, AND PAST LAST DATA LINE,
C***ALTER TO A BORDER LINE WITH SIDE SYMBOLS ONLY
      IF(K.EQ.5.AND.IOL.GT.IOLD) THEN
        K=6
        ILAP=1
        ML=0
      ENDIF
C***IF LAST LINE OF BOTTOM BORDER, ALTER TO FULL BORDER
C***LINE (WITHOUT NEW PAGE)
      IF(K.EQ.6.AND.ILAP.GE.NBBLK) THEN
        K=7
        ML=1
      ENDIF
C***IF A LINE OF BOTTOM BORDER PAST ITS LAST LINE, ALTER TO
C***AN END OF BLOCK DUMMY
      IF(K.EQ.7.AND.ILAP.GT.NBBLK) THEN
        K=8
        ML=8
      ENDIF
C***IF AN END OF BLOCK DUMMY AFTER THE LAST BLOCK TO BE
C***PROCESSED, ALTER TO AN END OF DATA DUMMY
      IF(K.EQ.8.AND.(IBLK.GE.NBLK.OR.(IBLK.GE.MBLK.AND.MBLK.GT.0)))
     +    THEN
        K=9
        ML=9
      ENDIF
C
C***IF MODE IS FROM EXISTING FILES, OR NO PROCESSING, SET
C***UNCONDITIONALLY TO END OF DATA DUMMY
      IF(MOD((MF/100),10).GE.8) THEN
        K=9
        ML=9
      ENDIF
C
      RETURN
      END
C*** OF PROSEQ
```

```
      SUBROUTINE PROLN (ML,LOL,LOL,LPL,JA,JB,JD,JO,JP,JQ)
C
C PROCESSES A LINE OF OUTPUT, REAL OR DUMMY, ACCORDING TO THE
C TYPE SPECIFIED BY ML. ARRAYS WITH ADJUSTABLE DIMENSIONS AND
C THEIR SIZES ARE GIVEN AS PARAMETERS. OTHER DATA IS PASSED IN
C COMMON. THIS ROUTINE IS CALLED FROM THE MAIN ROUTINE (EXPLOT).
C
      INTEGER JA(LDL),JB(LDL),JD(LDL),JO(LOL),JP(LPL),JQ(LPL)
      COMMON/STATUS/IEQ,IER,IES(32)
      COMMON/PROCOM/IBLK,KBLK,LBLK,MBLK,NBLK,NBBLK,NLPBLK,
     +              ISEC,KSEC,LSEC,MSEC,NSEC,NB    ,NLAP  ,
     +              IDL,IOL,IPL,NDL,NOL,NPL,
     +              MB,MF,MO,MP,MS,IIL,ILAP,KPL,KIDA,KIDB
      DATA KPCNP/1H1/
C
C***LOCAL MODE FOR OUTPUT METHOD. LESS THAN 3 IS IMMEDIATE
      MODE=MOD((MF/100),10)
C
C***PROCESS A LINE TO JP OR SECTION FILES
C
      IF(ML.LE.4) THEN
C  ***BORDER LINE TO BE GENERATED
        CALL LNBORD(JP,LPL,LOL,KSEC,LSEC,MSEC,NSEC,MB,MF,ML,KPL)
      ELSEIF(ML.LE.7) THEN
C  ***DATA LINE
C  ***REWIND SCRATCH FILE FOR FIRST LINE OF OVERLAP
        IF(ML.LE.6.AND.ILAP.EQ.1) REWIND 90
C  ***FIND INTERPOLATED POSITION IIL FOR THIS OUTPUT LINE
        CALL LNSPEC(IIL,IOL,NOL,NDL,MO)
C  ***GET OUTPUT DATA
        IF(ML.EQ.5) THEN
C    ***FOR TOP OVERLAP READ FROM SCRATCH FILE
          READ(90) (JO(I),I=1,LOL)
        ELSE
C    ***GET DATA FROM SOURCE, INTERPOLATE AND SCALE TO LENGTH
C    ***WITH RETURN FROM ERRORS
          CALL LNDATA(JA,JB,JD,LDL,IIL,MO,MS)
          IF(IER.GT.0) RETURN
          CALL LNSCAL(JO,LOL,JD,LDL,MO)
          IF(IER.GT.0) RETURN
        ENDIF
C  ***FOR BOTTOM OVERLAP COPY DATA LINE TO SCRATCH FILE
        IF(ML.EQ.6) WRITE(90) (JO(I),I=1,LOL)
C  ***FORM DATA INTO PRINTING LINE(S) IN JP OR SECTION FILES
        CALL LNOUT(JP,LPL,JO,LOL,KSEC,LSEC,MSEC,NSEC,
     +    MB,MF,IOL,IIL,ILAP,KPL)
      ENDIF
C
C***OUTPUT PLOTTING LINE(S)
C
      IF(MODE.LE.2) THEN
C  ***IMMEDIATE OUTPUT, PLOT JP OR WRITE A LABEL
        IF(ML.LE.7) CALL LNPLOT(JP,JQ,LPL,MF,MO,MP)
        IF(ML.GT.7) CALL BLKEND(MSEC,NSEC,IBLK,NBLK,
     +    KIDA,KIDB,MF,ML,MO,JP,LPL)
      ELSEIF((ML.GE.9.AND.(MODE.EQ.5.OR.MODE.EQ.8)).OR.
     +       (ML.GE.8.AND.MODE.EQ.4)) THEN
C
C  ***FROM SECTION FILES
C
C  ***OUTPUT AT END OF DATA OR END OF BLOCK DUMMY LINE
C  ***SET LIMITS AND LOOP OVER SECTIONS
        M=1
        N=NSEC
        IF(MSEC.GT.0) M=MSEC
        IF(MSEC.GT.0) N=MSEC
        DO 120 ISEC=M,N
C    ***SELECT AND REWIND SECTION FILE
          IU=ISEC+10
          REWIND IU
C    ***ZERO 'NEW PAGE CHARACTER' COUNTER
          INP=0
C    ***LOOP OVER LINES
          100 CONTINUE
C      ***IF OUT OF TIME, RETURN
            IF(SECOND(X).GE.FLOAT(IES(5))) RETURN
C      ***READ LINE AND TEST FIRST VALUE
            READ(IU,END=110) JP
            IF(JP(1).EQ.KPCNP) INP=INP+1
C      ***PLOT JP UNLESS FROM FILES, SINGLE BLOCK SELECTED,
C      ***AND NOT CORRECT BLOCK
            IF(MODE.NE.8.OR.MBLK.LE.0.OR.INP.EQ.MBLK)
     +        CALL LNPLOT(JP,JQ,LPL,MF,MO,MP)
          GO TO 100
          110 CONTINUE
C    ***REWIND SECTION FILE READY FOR NEXT BLOCK
          REWIND IU
C    ***ADD A LABEL
          CALL BLKEND(ISEC,NSEC,IBLK,NBLK,KIDA,KIDB,MF,ML,MO,JP,LPL)
        120 CONTINUE
      ENDIF
C
      RETURN
      END
C*** OF PROLN
```

```
      SUBROUTINE EXPLOT
C
C CAUSES PLOTTING ACCORDING TO THE PRESET COMMON VARIABLES.
C
      COMMON/LENGTH/LDL,LOL,LPL,LPPL
      COMMON/STATUS/IEQ,IER,IES(32)
      COMMON JM(150),JA(520),JB(520),JD(520),JO(520),JP(520),JQ(520)
C
      TIME=SECOND(X)
      CALL PRORDY(ML,LDL,LPL,LPPL,JA,JB,JD,JP)
      IF(IER.GT.0) RETURN
      WRITE(6,*) ' PLOTTING'
C
      N=0
  100 CONTINUE
        N=N+1
        CALL PROSEQ(ML)
        IF(IER.GT.0) RETURN
        CALL PROLN(ML,LDL,LOL,LPL,JA,JB,JD,JO,JP,JQ)
        IF(IER.GT.0) RETURN
      IF(ML.LT.9.AND.SECOND(X).LT.FLOAT(IES(5))) GO TO 100
C
      WRITE(6,1110) N,(SECOND(X)-TIME),IES(12)
 1110 FORMAT('1X,I5,' LINES, CP SECONDS',F7.2/
     +             6X,' PLOTFILE LENGTH NOW',I7)
C
      RETURN
      END
C*** OF EXPLOT
```

```
      SUBROUTINE NEWLN (MSS,MA,MX,J,L)
C
C SUPPLIES A LINE OF DATA.
C IF MSS +VE IT COMES FROM A FILE MOD(MSS,100) WITH ZERO
C DEFAULTING TO TAPE7.
C IF MSS -VE THE DATA IS INTERNALLY GENERATED DUMMY.
C IF THE THIRD DIGIT FROM THE RIGHT OF MSS IS NOT ZERO THE
C FILE IS NOT REWOUND DURING INITIALISATION.
C IF THE FOURTH DIGIT FROM THE RIGHT OF MSS IS NOT ZERO THE
C DATA IS SCALED BEFORE CONVERSION TO INTEGER BY 10**D.
C OTHERWISE BY 1000.
C THIS ROUTINE IS CALLED FROM LNDATA
C
      COMMON/STATUS/IEQ,IER,IES(29),MAXDAT,MAXPLT,NGRP
      DATA SC/1000.0/
      INTEGER J(L)
      REAL RD(257)
C
      MS=MOD(MSS,1000)
      IU=0
      IF(MS.GT.0) IU=MOD(MS,100)
      IF(IU.EQ.0) IU=7
C
      IF(MA.EQ.0) THEN
C  ***INITIALISE (DUMMY STATEMENT USING MX IS TO KILL A
C  ***COMPILER DIAGNOSTIC)
        LN=0
        IF(MX.EQ.0) LN=MX
        IF(MS.GE.0.AND.MS.LT.100) REWIND IU
C  ***SET SCALING FACTOR
        SC=1.0E3
        IF(MSS/1000.NE.0) SC=10.0**(MSS/1000)
      ELSE
        IF(MS.GE.0) READ(IU,END=80) (RD(I),I=1,L)
        GO TO 90
C     ***DATA NOT FOUND
   80   IER=7001
        IES(1)=LN
        IF(IEQ.GT.0) CALL ERROR
        RETURN
   90 CONTINUE
      LN=LN+1
      DO 100 I=1,L
        IF(MS.GE.0) THEN
          N=IFIX(SC*RD(I))
        ELSEIF(MA.LE.2) THEN
C         ***GENERATE DUMMY DATA (IF DATA IS TO BE OUTPUT)
          RL=FLOAT(L)
          RI=FLOAT(I-1)/RL
          RN=FLOAT(LN-1)/RL
          IF(MS.EQ.-1) THEN
            R=FLOAT(I)
          ELSEIF(MS.EQ.-2) THEN
            R=SC*RI
          ELSEIF(MS.EQ.-3) THEN
C           ***GAUSSIAN CENTRED ON TOP LEFT CORNER
            R=SC*EXP(-(RI**2+RN**2)*4.5)
          ELSEIF(MS.GE.-49) THEN
C           ***A DOUBLE STAR IMAGE
            A=-FLOAT(3+MS)
            RI=0.5+(RI-0.5)*A
            RN=0.5+(RN-0.5)*A
            R=SC*EXP(-((RI-0.333)**2+(RN-0.667)**2)*18.0)
            R=R+0.4*SC*EXP(-((RI-0.667)**2+(RN-0.333)**2)*18.0)
          ELSE
C           ***ARRAY FILED WITH A SINGLE VALUE
            R=-(SC/1000.0)*FLOAT(MS+100)
          ENDIF
          N=IFIX(R)
        ENDIF
C       ***COPY OR ADD INTO ARRAY ACCORDING TO MA
        IF(MA.EQ.1) J(I)=N
        IF(MA.EQ.2) J(I)=J(I)+N
  100 CONTINUE
C  ***TRAP OUT OF RANGE DATA
C  J(I)=MAXO(-MAXDAT,MINO(MAXDAT,J(I)))
      ENDIF
C
      RETURN
      END
C*** OF NEWLN
```

```
      SUBROUTINE BORDER (MB,NLAP,KSYM,MB,IOL,IIL,ILAP)
C
C RETURNS BORDER AND OVERLAP WIDTHS (SIDE), AND A SIDE BORDER
C SYMBOL DEPENDING ON LINE, SET BY MB. THIS ROUTINE IS CALLED
C FROM A NUMBER OF OTHER ROUTINES.
C
      DATA KBLANK,KA,KB,KC,KD,KE/1H ,1H+,1H=,1H*,1HO,1H-/
C
C***SET BORDER AND OVERLAP WIDTHS FROM MB
C
      MODE=MOD(MB,20)
      NB=MOD(MODE,5)
      IF(NB.EQ.3) NB=9
      IF(NB.EQ.2) NB=3
      NLAP=MODE/5
      IF(NLAP.EQ.3) NLAP=9
      IF(NLAP.EQ.2) NLAP=3
C
C***SET SYMBOL
C
C***ISOLATE SYMBOL MODE AND SET BASIC SYMBOL
      MODE=MOD((MB/100),10)
      KSYM=KA
C***IF UNMODIFIED LINES OF ORIGINAL DATA ARE TO BE MARKED
      IF(MODE.GE.8) THEN
        IF(IIL.GT.0.AND.MOD(IIL,128).EQ.0) KSYM=KB
      ENDIF
C***IF EVERY TENTH LINE IS TO BE MARKED
      IF(MODE.GE.6) THEN
        IF(MOD(IOL,10).EQ.0) KSYM=KC
      ENDIF
C***IF CENTRE OF OVERLAP IS TO BE MARKED
      IF(MODE.GE.4.AND.ILAP.GT.0) THEN
        IF(NLAP.EQ.1.AND.ILAP.EQ.1) THEN
          KSYM=KD
        ELSEIF(NLAP.GT.1) THEN
          IF(MOD(MODE,2).EQ.0) THEN
C           ***SINGLE SYMBOL
            IF(ILAP.EQ.((NLAP+1)/2)) KSYM=KD
          ELSE
C           ***PAIR OF SYMBOLS
            IF(ILAP.EQ.((NLAP+1)/2-1)) KSYM=KD
            IF(ILAP.EQ.((NLAP+1)/2+1)) KSYM=KD
          ENDIF
        ENDIF
      ENDIF
C
C***MODIFY SYMBOL
C
      MODE=MOD((MB/20),5)
      IF(MODE.EQ.1.AND.KSYM.EQ.KA) KSYM=KBLANK
      IF(MODE.EQ.2.AND.KSYM.EQ.KD) KSYM=KBLANK
      IF(MODE.EQ.3.AND.KSYM.EQ.KD) KSYM=KE
      IF(MODE.EQ.4) KSYM=KBLANK
C
      RETURN
      END
C*** OF BORDER
C
C
C
      SUBROUTINE ERROR
C
C PRINTS, TO UNIT 6, THE ERROR TERMINATION NUMBER ON A NEW
C PAGE, FOLLOWED BY THE STATUS VARIABLES, AND STOPS. IT CAN
C BE CALLED FROM ANY ROUTINE WHEN AN ERROR IS DETECTED, IF
C IEQ=1.
C
      COMMON/STATUS/IEQ,IER,IES(32)
C
      WRITE(6,1100) IER
 1100 FORMAT(1H1/19H ERROR TERMINATION ,I4)
      WRITE(6,1110)IES
 1110 FORMAT(7H STATUS,8I7,3(/7X,8I7)/1H1)
      IF(IEQ.EQ.1) STOP
      RETURN
      END
C*** OF ERROR
```

```
      SUBROUTINE BLKEND
     +  (ISEC,NSEC,IBLK,NBLK,KIDA,KIDB,MF,ML,MO,JP,LPL)
C
C PRODUCES A LABEL LINE AFTER A BLOCK OF DATA HAS BEEN PLOTTED.
C THE FIRST SIX PARAMETERS PROVIDE THE TEXT. MF GIVES THE
C TYPE OF LABEL AND DESTINATION (EITHER OF TWO PRINT FILES OR
C THE MERGE FILE) ALONG WITH MO INDICATING MICROFILM OR
C PRINTER. FOR MICROFILM A CALL IS MADE TO MFEND. ML INDICATES
C END OF BLOCK OR END OF DATA AND ALONG WITH THE FIRST FOUR
C PARAMETERS CONTROLS THE PRODUCTION OF A LABEL. THIS
C ROUTINE IS CALLED FROM PROLN.
C
      INTEGER JT(2),JL(11),JP(LPL)
      DATA KBLANK/1H /
C
C***ISOLATE OUTPUT AND LABEL TYPE PARAMETERS
      MODE=MOD((MO/100),10)
      M=MOD(MF,10)
C
      IF(MODE.LE.4) THEN
C       ***TO PRINTER
        IF(MOD((MF/100),10).NE.1) THEN
C         ***NOT MERGED
C         ***SELECT DESTINATION FILE
          IU=10
          IF(MOD((MF/2000),2).EQ.1) IU=9
C         ***WRITE LABEL
          IF(M.EQ.1.AND.ML.EQ.9.AND.ISEC.EQ.NSEC)
     +      WRITE(IU,1100) KIDA,KIDB
 1100     FORMAT(1H2,20X,14H***END OF PLOT,2I5,4H ***)
          IF(M.EQ.2.AND.ML.EQ.9)
     +      WRITE(IU,1110) ISEC,NSEC,KIDA,KIDB
 1110     FORMAT(1H2,20X,13H***END OF SEC,I3,3H OF,I3,
     +      6H, PLOT,2I5,4H ***)
          IF(M.EQ.3.AND.ISEC.EQ.NSEC)
     +      WRITE(IU,1120) IBLK,NBLK,KIDA,KIDB
 1120     FORMAT(1H2,20X,13H***END OF BLK,I3,3H OF,I3,
     +      6H, PLOT,2I5,4H ***)
          IF(M.EQ.4) WRITE(IU,1130) IBLK,NBLK,ISEC,NSEC,KIDA,KIDB
 1130     FORMAT(1H2,20X,6H***BLK,I3,3H OF,I3,5H, SEC,
     +      I3,3H OF,I3,6H, PLOT,2I5,4H ***)
          IF(M.EQ.5) WRITE(IU,1140) KIDA,KIDB
 1140     FORMAT(1HO,2I5,///)
C
        ELSEIF(M.EQ.1) THEN
C         ***MERGED, AND LABEL REQUIRED
C         ***CONVERT TO CHARACTER STRING
          ENCODE(11,1150,JT(1)) KIDA,KIDB
 1150     FORMAT(1HO,2I5)
          DECODE(11,1160,JT(1)) JL
 1160     FORMAT(11A1)
C         ***FILL JP
          DO 170 IPP=1,LPL
            JP(IPP)=KBLANK
            IF(IPP.LE.11) JP(IPP)=JL(IPP)
  170     CONTINUE
C         ***OUTPUT TO MERGE FILE
          CALL LMERGE(JP,LPL)
        ENDIF
C
      ELSE
C       ***OUTPUT TO MICROFILM
C       ***SET TYPE REQUIRED
        K=0
        IF(MOD((MF/100),10).NE.1) THEN
C         ***NOT MERGED
          IF(M.EQ.1.AND.ML.EQ.9.AND.ISEC.EQ.NSEC) K=M
          IF(M.EQ.2.AND.ML.EQ.9) K=M
          IF(M.EQ.3.AND.ISEC.EQ.NSEC) K=M
          IF(M.EQ.4.OR.M.EQ.5) K=M
        ELSE
C         ***MERGED
          K=10
        ENDIF
C       ***OUTPUT MICROFILM LABEL IF REQUIRED
        IF(K.NE.0) CALL MFEND(ISEC,NSEC,IBLK,NBLK,KIDA,KIDB,K)
      ENDIF
C
      RETURN
      END
C*** OF BLKEND
```

```
      SUBROUTINE LNSPEC (IIL,IOL,NOL,NDL,MO)
C
C RETURNS FOR OUTPUT LINE NUMBER IOL (OF NOL OUTPUT LINES) A
C POSITION IIL AMONG THE NDL INPUT DATA LINES, EXPRESSED IN
C UNITS OF 1/128. IT IS CALCULATED ACCORDING TO THE METHOD
C SPECIFIED BY MO, AND THE SIGN OF IIL INDICATES THE WAY THE
C OUTPUT LINE SHOULD BE GENERATED. +VE IS INTERPOLATED,
C -VE IS GROUPED OR INTEGRATED (INTEGRATED NOT PRESENTLY
C IMPLEMENTED). THIS ROUTINE IS CALLED FROM PRORDY AND PROLN.
C
C***FIND LOCAL MODE FROM MO, MODIFIED BY NOL,NDL
C
      MODE=MOD((MO/10),10)
      IF(MODE.GT.4) MODE=0
      IF(MODE.EQ.0) MODE=3
      IF(NOL.EQ.NDL) MODE=10
      IF((NOL.LT.NDL).AND.(MODE.LE.3)) MODE=11
      IF(((7*NOL).LE.(4*NDL)).AND.(MODE.EQ.4)) MODE=11
C
C
      IF(MODE.EQ.1) THEN
C  ***REPEAT INPUT LINES
        IIL=((((IOL-1)*NDL)/NOL)+1)*128
      ELSEIF(MODE.EQ.10) THEN
C  ***COPY INPUT LINES
        IIL=IOL*128
      ELSEIF(MODE.EQ.11) THEN
C  ***GROUP AND AVERAGE INPUT LINES
        IIL=-(((IOL*NDL)/NOL)*128+64)
      ELSE
C  ***INTERPOLATE BETWEEN INPUT LINES
C  ***LOWER BOUND OF INTERVAL, AND OF NEXT, CENTRE OF INTERVAL
        A=((FLOAT(IOL)-1.5)*FLOAT(NDL-1)/FLOAT(NOL-1))+1.0
        B=((FLOAT(IOL+1)-1.5)*FLOAT(NDL-1)/FLOAT(NOL-1))+1.0
        C=(A+B)/2.0
        IF(MODE.EQ.2) THEN
C         ***ORIGINAL INPUT LINES ONCE EACH, INTERPOLATES MIDPOINT
          IF(INT(A).NE.INT(B)) THEN
            IIL=INT(B)*128
          ELSE
            IIL=INT(B)*128+64
          ENDIF
        ELSEIF(MODE.EQ.3) THEN
C         ***ORIGINAL INPUT LINES ONCE EACH, INTERPOLATES EXACT
          IF(INT(A).NE.INT(B)) THEN
            IIL=INT(B)*128
          ELSE
            IIL=INT(C*128.0+0.5)
          ENDIF
        ELSEIF(MODE.EQ.4) THEN
C         ***EXACT INTERPOLATES
          IIL=INT(C*128.0+0.5)
        ENDIF
      ENDIF
C
      RETURN
      END
C*** OF LNSPEC
C
C
C
      SUBROUTINE LNDATA (JA,JB,JD,LDL,IIL,MO,MS)
C
C PLACES IN JD A LINE OF DATA DERIVED ACCORDING TO THE
C INTERPOLATED POSITION IIL FROM THE INPUT DATA ACCESSED BY
C NEWLN. FOR IIL=0 THE SOURCE IS INITIALISED. JA, JB ARE USED
C FOR LOCAL STORAGE OF LINES OF INPUT DATA. MS ALLOWS DATA
C SOURCE TO BE SELECTED, MO CAN SPECIFY HIGHER ORDER
C INTERPOLATION (NOT IMPLEMENTED). THIS ROUTINE IS CALLED
C FROM PROLN.
C
      INTEGER JA(LDL),JB(LDL),JD(LDL)
      COMMON/STATUS/IEQ,IER,IES(32)
C
      IF(IIL.EQ.0) THEN
C  ***INITIALISE DATA SOURCE
        CALL NEWLN(MS,0,0,JD,LDL)
C  ***DUMMY STATEMENT TO SUPRESS COMPILER DIAGNOSTIC
        IF(MO.EQ.0) IILAST=IIL
C
      ELSEIF(IIL.EQ.IILAST) THEN
C  ***SAME AS PREVIOUS CALL. NO ACTION
C
      ELSEIF(IIL.LT.0) THEN
C  ***AVERAGE GROUP OF LINES OR INTEGRATE THEM
C
        IF(MOD(-IIL,128).NE.64) THEN
C         ***INTEGRATE (UNIMPLEMENTED). ERROR
          IER=5001
        ELSE
C         ***AVERAGE OVER GROUP OF LINES
C         ***NUMBER TO BE AVERAGED. TEST FOR OUT OF SEQUENCE
C         ***OR EXCESSIVE
          N=IABS(IIL)/128-IABS(IILAST)/128
          IF(N.LT.1) THEN
            IER=5002
          ELSEIF(N.GT.5000) THEN
            IER=5003
          ELSE
C           ***GET FIRST LINE
            CALL NEWLN(MS,1,0,JD,LDL)
C           ***ADD ANY MORE LINES AND DIVIDE BY TOTAL NUMBER
            IF(N.GE.2) THEN
              DO 110 I=2,N
                CALL NEWLN(MS,2,0,JD,LDL)
              110 CONTINUE
              DO 120 J=1,LDL
                JD(J)=JD(J)/N
              120 CONTINUE
            ENDIF
          ENDIF
        ENDIF
C
      ELSEIF(IIL.GT.0) THEN
C  ***INTERPOLATE BETWEEN LINES
C
C  ***CALCULATE ADVANCE THROUGH DATA LINES
        N=(IIL+127)/128-(IILAST+127)/128
        IF(N.LT.0) THEN
C         ***LINES ACCESSED OUT OF ORDER, ERROR
          IER=5004
        ELSEIF(N.GT.5000) THEN
C         ***EXCESSIVE LINE ADVANCE, ERROR
          IER=5005
        ELSEIF(N.EQ.0) THEN
C         ***NO ADVANCE NEEDED
        ELSEIF(N.EQ.1) THEN
C         ***ADVANCE ONE LINE, COPYING JB INTO JA
          DO 130 J=1,LDL
            JA(J)=JB(J)
          130 CONTINUE
          CALL NEWLN(MS,1,0,JB,LDL)
        ELSEIF(N.EQ.2) THEN
          CALL NEWLN(MS,1,0,JA,LDL)
          CALL NEWLN(MS,1,0,JB,LDL)
        ELSEIF(N.GT.2) THEN
          DO 140 I=1,N-2
            CALL NEWLN(MS,3,0,JA,LDL)
          140 CONTINUE
          CALL NEWLN(MS,1,0,JA,LDL)
          CALL NEWLN(MS,1,0,JB,LDL)
        ENDIF
C
C  ***IF NO ERROR, GENERATE INTERPOLATED LINE IN JD
        IF(IER.LE.0) THEN
C         ***FRACTIONAL POSITION
          II=MOD(IIL,128)
          IF(II.EQ.0) THEN
C           ***COPY JB TO JD
            DO 150 J=1,LDL
              JD(J)=JB(J)
            150 CONTINUE
          ELSEIF(II.EQ.64) THEN
C           ***SET JD TO AVERAGE OF JA, JB
            DO 160 J=1,LDL
              JD(J)=(JA(J)+JB(J))/2
            160 CONTINUE
          ELSE
C           ***JD IS II/128 FROM JA TO JB
            JJ=128-II
            DO 170 J=1,LDL
              JD(J)=(JJ*JA(J)+II*JB(J)+64)/128
            170 CONTINUE
          ENDIF
        ENDIF
      ENDIF
C
C***UPDATE RECORD OF LAST CALL, AND CALL ERROR IF NEEDED
C
      IILAST=IIL
      IF(IER.GT.0.AND.IEQ.GT.0) CALL ERROR
C
      RETURN
      END
C*** OF LNDATA
```

```
      SUBROUTINE LNSCAL (JO,LOL,JD,LDL,MO)
C
C TAKES DATA IN JD AND ADJUSTS IT IN LENGTH TO FIT JO
C USING THE METHOD INDICATED BY MO. IF LOL IS NOT
C CORRECTLY RELATED TO LDL IT IS TREATED AS AN ERROR.
C ACCEPTABLE ARE NOMINAL INTEGRAL RATIOS EITHER WAY, 3/2
C AND 2/3. THIS ROUTINE IS CALLED FROM PROLN.
C
      INTEGER JO(LOL),JD(LDL)
      COMMON/STATUS/IEQ,IER,IES(32)
C
C***FIND LOCAL MODE AND SUBSTITUTE DEFAULTS
      MODE=MOD(MO,10)
      IF(MODE.GT.4) MODE=0
      IF(MODE.EQ.0) MODE=4
C
C
      IF(LOL.EQ.LDL) THEN
C  ***LINES OF EQUAL LENGTH, COPY JD TO JO
C
        DO 100 I=1,LOL
          JO(I)=JD(I)
  100   CONTINUE
C
      ELSEIF(LOL.LT.LDL) THEN
C  ***REDUCE LENGTH
C
        IF(MOD(LDL,LOL).LT.(LDL/LOL)) THEN
C     ***CLOSE ENOUGH TO INTEGER RATIO. AVERAGE IN GROUPS
          N=LDL/LOL
          M=0
          DO 120 I=1,LOL
            JTEMP=0
            DO 110 J=1,N
              M=M+1
              JTEMP=JTEMP+JD(M)
  110       CONTINUE
            JO(I)=JTEMP/N
  120     CONTINUE
        ELSEIF(LOL.EQ.((2*LDL)/3)) THEN
C     ***FITS 3/2 REDUCTION. OUTPUT ALTERNATES ORIGINAL
C     ***VALUES WITH AVERAGE OF PAIRS
          M=0
          DO 130 I=1,LOL
            IF(MOD(I,2).NE.0) THEN
              M=M+2
              JO(I)=(JO(M-1)+JO(M))/2
            ELSE
              M=M+1
              JO(I)=JD(M)
            ENDIF
  130     CONTINUE
        ELSE
C     ***INEXACT RATIO, FLAG LOCAL MODE
          MODE=MODE+20
        ENDIF
C
      ELSEIF(LOL.GT.LDL) THEN
C  ***INCREASE LENGTH
C
        IF(MODE.EQ.1) THEN
C     ***REPEAT VALUES
          IF(MOD(LOL,LDL).EQ.0) THEN
C       ***EXACT INTEGER RATIO
            N=LOL/LDL
            M=0
            DO 150 I=1,LDL
              DO 140 J=1,N
                M=M+1
                JO(M)=JD(I)
  140         CONTINUE
  150       CONTINUE
          ELSEIF(LOL.EQ.((3*LDL)/2)) THEN
C       ***EXACT 3/2 RATIO
            M=0
            DO 160 I=1,LOL
              IF(MOD(I,3).NE.0) M=M+1
              JO(I)=JD(M)
  160       CONTINUE
          ELSE
C       ***INEXACT RATIO, FLAG LOCAL MODE
            MODE=MODE+20
          ENDIF
C
        ELSEIF(MODE.GE.2) THEN
C     ***INTERPOLATE
          IF(MOD((LOL-1),(LDL-1)).EQ.0) THEN
C       ***EXACT INTEGER RATIO, N
            N=(LOL-1)/(LDL-1)
            IF(MODE.EQ.2.OR.N.EQ.2) THEN
C         ***ONLY MIDPOINT INTERPOLATES
              M=0
              DO 180 I=1,LDL-1
                M=M+1
                JO(M)=JD(I)
                DO 170 J=1,N-1
                  M=M+1
                  JO(M)=(JD(I)+JD(I+1))/2
  170           CONTINUE
  180         CONTINUE
            ELSE
C         ***EXACT INTERPOLATES
              M=0
              DO 200 I=1,LDL-1
                M=M+1
                JO(M)=JD(I)
                DO 190 J=1,N-1
                  M=M+1
                  JO(M)=((N-J)*JD(I)+J*JD(I+1))/N
  190           CONTINUE
  200         CONTINUE
            ENDIF
            JO(LOL)=JD(LDL)
          ELSEIF(LOL.EQ.((3*(LDL-1))/2+1)) THEN
C       ***EXACT 3/2 RATIO
            IF(MODE.LE.3) THEN
C         ***ORIGINAL VALUES AND INTERPOLATES
              M=0
              DO 210 I=1,LOL
                IF(MOD(I,3).NE.0) THEN
                  M=M+1
                  JO(I)=JD(M)
                ELSE
                  JO(I)=(JD(M)+JD(M+1))/2
                ENDIF
  210         CONTINUE
            ELSEIF(MODE.EQ.4) THEN
C         ***EXACT INTERPOLATION
              M=0
              DO 220 I=1,LOL
                IF(MOD(I,3).EQ.1) THEN
                  M=M+1
                  JO(I)=JD(M)
                ELSEIF(MOD(I,3).EQ.2) THEN
                  M=M+1
                  JO(I)=(JD(M-1)+2*JD(M))/3
                ELSE
                  JO(I)=(2*JD(M)+JD(M+1))/3
                ENDIF
  220         CONTINUE
            ENDIF
          ELSE
C       ***INEXACT RATIO, FLAG LOCAL MODE
            MODE=MODE+20
          ENDIF
        ENDIF
      ENDIF
C
C***INEXACT RATIO ERROR HANDLING
C
      IF(MODE.GT.20) THEN
        IER=5101
        IF(IEQ.GT.0) CALL ERROR
      ENDIF
C
      RETURN
      END
C*** OF LNSCAL
```

```
      SUBROUTINE LNBORD (JP,LPL,LOL,
     +                   KSEC,LSEC,MSEC,NSEC,MB,MF,ML,KPL)
C
C PRODUCES ONE OR MORE OUTPUT BORDER LINES OF TYPE SPECIFIED
C BY ML AND ACCORDING TO THE SECTIONING SPECIFICATION KSEC...
C IT WRITES THEM TO THE SECTION FILES, OR LEAVES ONE IN JP.
C THIS ROUTINE IS CALLED FROM PROLN.
C
      INTEGER JP(LPL)
      DATA KPC,KPCNP/1H ,1H1/
      DATA KBLANK,KA,KC,KD,KE,KBT,KB8/1H ,1H+,1H*,1H0,1H-,2H+T,2H+8/
C
C***SET BORDER PARAMETERS
C
      CALL BORDER(MB,NLAP,KSYM,MB,1,0,0)
C
C***SELECT BORDER SYMBOLS, RESET LOCAL MODE FOR MARKINGS
C
      MODE=MOD((MB/20),5)
      KS=KA
      KLAP=KD
      IF(MODE.EQ.1.OR.MODE.EQ.4) KS=KBLANK
      IF(MODE.EQ.2) KLAP=KBLANK
      IF(MODE.EQ.3) KLAP=KE
      MODE=MOD((MB/100),10)
C
C***SET LOOP LIMITS FOR SECTIONS TO BE OUTPUT
C***MSEC=0 IS ALL SECTIONS, +VE IS SECTION MSEC
C
      M=1
      N=NSEC
      IF(MSEC.GT.0) M=MSEC
      IF(MSEC.GT.0) N=MSEC
C
C***LOOP OVER SECTIONS TO BE OUTPUT
C
      DO 140 ISEC=M,N
C  ***LIMITS OF DATA USED IN CORRESPONDING DATA OUTPUT LINE
        IA=(ISEC-1)*LSEC+KSEC+1
        IB=IA+(LSEC-1)+NLAP
        IA=MINO(IA,LOL)
        IB=MINO(IB,LOL)
C  ***TOTAL OUTPUT LINE LENGTH TO END OF BORDER
        LT=IB-IA+2+2*MB+KPL
        LT=MINO(LT,LPL)
C  ***SET POINTER TO HYPOTHETICAL DATA POINT BEFORE BORDER
        IOP=IA-MB-1
C
        IF(MOD(ML,2).EQ.0) THEN
C    ***A BASICALLY BLANK BORDER LINE, ML 0,2,4
          DO 100 IPP=2,LPL
            JP(IPP)=KBLANK
  100     CONTINUE
        ELSE
C    ***A FULL BORDER LINE, ML 1,3
C    ***BLANK FILL ANY OFFSET
          IF(KPL.GT.0) THEN
            DO 110 IPP=2,KPL+1
              JP(IPP)=KBLANK
  110       CONTINUE
          ENDIF
C    ***LOOP OVER BORDER POINTS
          DO 120 IPP=KPL+2,LT
            JP(IPP)=KS
C      ***MARK EVERY TENTH DATA POSITION IF REQUIRED
            IOP=IOP+1
            IF(MODE.GE.6.AND.IOP.GE.IA.AND.IOP.LE.IB) THEN
              IF(MOD(IOP,10).EQ.0) JP(IPP)=KC
            ENDIF
  120     CONTINUE
C    ***BLANK FILL ANY REMAINING LENGTH
          IF(LT.LT.LPL) THEN
            DO 130 IPP=LT+1,LPL
              JP(IPP)=KBLANK
  130       CONTINUE
          ENDIF
C    ***MARK CENTRE OF OVERLAP IF REQUIRED
          IF(MODE.GE.4) THEN
            IF(NLAP.EQ.1) THEN
              JP(2+KPL+NB)=KLAP
              JP(LT-NB)=KLAP
            ELSEIF(NLAP.GT.1) THEN
C        ***CENTRE POINTS OF FIRST, LAST OVERLAPS
              ILAPA=1+NB+(1+NLAP)/2+KPL
              ILAPB=LT-NB-NLAP+(1+NLAP)/2
              IF(ILAPA.GT.1.AND.ILAPB.GT.1) THEN
                IF(MOD(MODE,2).EQ.0) THEN
C                 ***SINGLE OVERLAP MARKER SYMBOL
                  JP(ILAPA)=KLAP
                  JP(ILAPB)=KLAP
                ELSE
C                 ***DOUBLE OVERLAP MARKER SYMBOL
                  JP(ILAPA-1)=KLAP
                  JP(ILAPA+1)=KLAP
                  JP(ILAPB-1)=KLAP
                  JP(ILAPB+1)=KLAP
                ENDIF
              ENDIF
            ENDIF
          ENDIF
        ENDIF
C
C  ***SET SIDE SYMBOLS
        KSIDE=KS
        IF((MODE/2).EQ.1.AND.(ML.EQ.1.OR.ML.EQ.3)) KSIDE=KA
C  ***BOX BORDER
        IF(MOD(MB/20,5).EQ.4) THEN
          IF(ML.EQ.1) KSIDE=KB8
          IF(ML.EQ.3) KSIDE=KBT
        ENDIF
        IF(ML.EQ.0.OR.ML.EQ.1.OR.ML.EQ.3) THEN
          JP(KPL+2)=KSIDE
          JP(LT)=KSIDE
        ENDIF
C  ***SET PRINTER CONTROL CHARACTER
        JP(1)=KPC
        IF(ML.GE.3) JP(1)=KPCNP
C
C  ***WRITE TO SECTION FILE UNLESS IMMEDIATE OUTPUT
        IU=ISEC+10
        IF(MOD((MF/100),10).GT.2) WRITE(IU) JP
C
  140 CONTINUE
C
      RETURN
      END
C*** OF LNBORD
```

```
      SUBROUTINE LNOUT (JP,LPL,JQ,LOL,
     +    KSEC,LSEC,MSEC,NSEC,MB,MF,IOL,IIL,ILAP,KPL)
C
C PRODUCES ONE OR MORE OUTPUT LINES ACCORDING TO THE
C SECTIONING SPECIFICATION KSEC...(COMPLETE WITH OVERLAP
C AND BORDER IF REQUIRED) FROM JQ, AND WRITES THEM TO
C THE SECTION FILES, OR LEAVES ONE IN JP. THIS ROUTINE
C IS CALLED FROM PROLN.
C
      INTEGER JP(LPL),JQ(LOL)
      DATA KPC,KBLANK/10H ++++++++++,10H          /
C
C***SET BORDER PARAMETERS
C
      CALL BORDER(NB,NLAP,KSYM,MB,IOL,IIL,ILAP)
C
C***SET LOOP LIMITS FOR SECTIONS TO BE PROCESSED
C***MSEC=0 IS ALL SECTIONS, +VE IS SECTION MSEC
C
      M=1
      N=NSEC
      IF(MSEC.GT.0) M=MSEC
      IF(MSEC.GT.0) N=MSEC
C
C***LOOP OVER SECTIONS TO BE OUTPUT
C
      DO 130 ISEC=M,N
C  ***SET POINTER TO FIRST POSITION
        IPP=1
C  ***SET PRINTER CONTROL VALUE
        JP(1)=KPC
C  ***BLANK FILL ANY LEADING OFFSET AND BORDER IN JP
        KL=KPL+NB
        IF(KL.GT.0) THEN
          DO 100 I=1,KL
            IPP=IPP+1
            JP(IPP)=KBLANK
  100     CONTINUE
        ENDIF
C
C  ***LIMITS OF DATA TO BE USED
        IA=(ISEC-1)*LSEC+KSEC+1
        IB=IA+(LSEC-1)+NLAP
        IA=MINO(IA,LOL)
        IB=MINO(IB,LOL)
C  ***COPY DATA INTO JP
        DO 110 I=IA,IB
          IPP=IPP+1
          IF(IPP.LE.LPL) JP(IPP)=JQ(I)
  110   CONTINUE
C
C  ***BLANK FILL ANY REMAINING SPACE
        IF(IPP.LT.LPL) THEN
          DO 120 I=IPP+1,LPL
            JP(I)=KBLANK
  120     CONTINUE
        ENDIF
C  ***SET ANY BORDER SYMBOL
        IF(NB.GT.0) THEN
          JP(KPL+2)=KSYM
          IF((IPP+NB).LE.LPL) JP(IPP+NB)=KSYM
        ENDIF
C
C  ***WRITE TO SECTION FILE UNLESS IMMEDIATE OUTPUT MODE
        IF(MOD((MF/100),10).GT.2) THEN
          IU=ISEC+10
          WRITE(IU) JP
        ENDIF
  130 CONTINUE
C
      RETURN
      END
C*** OF LNOUT
```

```
      SUBROUTINE LNPLOT (JP,JQ,LPL,MF,MO,MP)
C
C OUTPUTS A LINE OF DATA FROM JP IN MODE SPECIFIED BY MO.
C JQ IS USED BY THE CONTOURING ROUTINES FOR STORING THE
C PREVIOUS LINE OF DATA. THIS ROUTINE IS CALLED FROM PROLN.
C
      INTEGER JP(LPL),JQ(LPL)
      COMMON/STATUS/IEQ,IER,IES(32)
      DATA KBLANK,KPCNP/1H ,1H1/
C
C***PROCESS NEW PAGE ACCORDING TO NEW PAGE FLAG IES(3)
C
      MODE=MOD((MO/100),10)
      IF(JP(1).EQ.KPCNP) THEN
        IF(IES(3).EQ.0) THEN
          IF(MODE.GE.5) CALL MFPAGE(MF,MO,MP,LPL)
        ELSE
          JP(1)=KBLANK
          IES(3)=0
        ENDIF
      ENDIF
C
C***PLOT JP
C
      IF(MODE.EQ.0) CALL LPRTZ(JP,LPL,MF)
      IF(MODE.EQ.2) CALL LDMP(JP,LPL,MF)
      IF(MODE.EQ.5) CALL MFZ(JP,LPL,MP)
      IF(MODE.EQ.6) CALL MFC(JP,JQ,LPL,MP)
C
      RETURN
      END
C*** OF LNPLOT
```

```
      SUBROUTINE LPRTZ (JP,LPL,MF)
C
C CONVERTS THOSE VALUES IN JP WHICH REPRESENT DATA INTO VALUES
C FROM KONSYM, EACH REPRESENTING ONE OR MORE PRINTING SYMBOLS
C TO BE SUPERIMPOSED. JP IS THEN PASSED TO LPRT, WITH THE
C NUMBER OF OVERPRINTED OUTPUT LINES SET TO SUIT KONSYM, OR 1
C IF NO VALUES REPRESENTED DATA. VALUES OUTSIDE THE RANGE OF
C DATA ARE PASSED UNALTERED. THIS ROUTINE IS CALLED FROM
C LNPLOT.
C
      INTEGER JP(LPL)
      COMMON/KON/K(128),KONLEV(128),KONSYM(128),KONTAB(2048)
      COMMON/STATUS/IEQ,IER,IES(29),MAXDAT,MAXPLT,NGRP
C
C***ZERO DATA FLAG
      IDATA=0
C
C***LOOP OVER POSSIBLE DATA POSITIONS
C
      DO 100 IPP=2,LPL
        J=JP(IPP)
C  ***IF DATA VALUE, CONVERT TO SYMBOL
        IF(IABS(J).LE.MAXDAT) THEN
C    ***SET DATA FLAG
          IDATA=1
C    ***INITIAL SCALING, LINEAR OR LOG
          IF(K(9).EQ.0) THEN
            J=(2*K(6)*(J-K(2)))/K(3)
          ELSE
            J=IFIX(FLOAT(K(6))*
     +          ALOG10(AMAX1(1.E-6,(FLOAT(J-K(2))/FLOAT(K(3))))))
          ENDIF
C    ***LIMIT, INVERT OR LOOK UP TABLE AS REQUIRED
          J=MAXO(0,MINO(MAXPLT,J))
          IF(K(11).NE.0) J=MAXPLT-J
          IF(K(10).NE.0) J=KONTAB(J+1)
C    ***CONVERT TO LEVEL NUMBER, LIMIT, LOOK UP SYMBOL TABLE
          J=((J/NGRP)-K(7))/K(8)
          J=MAXO(0,MINO(127,J))
          JP(IPP)=KONSYM(J+1)
        ENDIF
  100 CONTINUE
C
C***SET NUMBER OF OVERPRINTED OUTPUT LINES AND HAVE THEM OUTPUT
      NP=1
      IF(IDATA.EQ.1) NP=K(11)
      CALL LPRT(JP,LPL,MF,NP)
C
      RETURN
      END
C*** OF LPRTZ
C
C
C
      SUBROUTINE LPRT (JP,LPL,MF,NP)
C
C OUTPUTS THE DATA IN JP TO A PRINT FILE SELECTED BY MF AS NP
C OVERPRINTED LINES, EACH LINE TAKING THE LEFTMOST REMAINING
C CHARACTER FROM EACH ELEMENT OF JP. IN MERGE MODE EACH OUTPUT
C LINE IS PASSED INSTEAD TO LMERGE FOR INCORPORATION IN THE
C MERGE FILE. THIS ROUTINE IS CALLED FROM LPRTZ AND LPRTC.
C
      INTEGER JP(LPL)
      COMMON/STATUS/IEQ,IER,IES(32)
C
C***SET DESTINATION, LIMIT NUMBER OF OVERPRINTS
      IU=10
      IF(MOD((MF/2000),2).EQ.1) IU=9
      N=MINO(NP,6)
C
C***LOOP OVER OUTPUT LINES
C
      DO 120 I=1,N
C  ***SHIFT ONE PLACE LEFT EXCEPT ON FIRST LINE
        IF(I.NE.1) THEN
          DO 100 IPP=1,LPL
            JP(IPP)=SHIFT(JP(IPP),6)
  100     CONTINUE
        ENDIF
C  ***OUTPUT LINE
        IF(MOD((MF/100),10).EQ.1) THEN
          CALL LMERGE(JP,LPL)
        ELSE
          WRITE(IU,1110) JP
 1110     FORMAT(137A1)
C    ***INCREMENT LINE COUNT
          IES(IU)=IES(IU)+1
        ENDIF
  120 CONTINUE
C
      RETURN
      END
C*** OF LPRT
      SUBROUTINE LMERGE (JP,LPL)
C
C READS A LINE FROM (91) AND COPIES TO (92) WITH JP APPENDED.
C FILES ARE READ A LINE AT A TIME, AND THE LAST VALUE GIVES
C THE NUMBER OF POSITIONS ALREADY USED. IF A LINE IS EMPTY
C (NUMBER=1) THE PRINTER CONTROL VALUE IS COPIED INTO THE
C FIRST PLACE, OTHERWISE IT IS DISCARDED. SOURCE FILE IS
C REWOUND WHEN IT REACHES THE END, AN EMPTY BLANK LINE
C BEING SUBSTITUTED. THIS ROUTINE IS CALLED FROM LPRT AND PROLN.
C
      INTEGER JP(LPL)
      COMMON JM(150)
      DATA KBLANK/1H /
C
C***READ A LINE. REWIND AND SUPPLY EMPTY BLANK LINE AT EOF
C
      READ(91,END=100) JM
      GO TO 120
  100 CONTINUE
      DO 110 I=1,150
  110 JM(I)=KBLANK
      JM(150)=1
      REWIND 91
  120 CONTINUE
C
C***COPY JP INTO LINE IF NOT FULL
C
      IF(JM(150).LT.149) THEN
        IF(JM(150).LE.1) JM(1)=JP(1)
        IA=JM(150)+1
        IB=MINO(149,(IA+LPL-2))
        J=1
        DO 130 I=IA,IB
          J=J+1
          JM(I)=JP(J)
  130   CONTINUE
        JM(150)=IB
      ENDIF
C
      WRITE(92) JM
C
      RETURN
      END
C*** OF LMERGE
C
C
C
      SUBROUTINE LDMP (JP,LPL,MF)
C
C OUTPUTS THE DATA IN JP TO A PRINT FILE SELECTED BY MF, AS
C INTEGERS. VALUES OUTSIDE THE DATA RANGE ARE CONVERTED TO
C SMALL INTEGERS. THE FIRST VALUE IS TRANSMITTED IN A1
C FORMAT FOR PRINTER CONTROL. THIS ROUTINE IS CALLED FROM
C LNPLOT.
C
      INTEGER JP(LPL)
      COMMON/STATUS/IEQ,IER,IES(29),MAXDAT,MAXPLT,NGRP
      DATA KBLANK/1H /
C
C***SELECT PRINT FILE
      IU=10
      IF(MOD((MF/2000),2).EQ.1) IU=9
C
C***LOOP OVER POSSIBLE DATA POSITIONS, CONVERT NON-DATA VALUES
C
      DO 100 IPP=2,LPL
        IF(IABS(JP(IPP)).GT.MAXDAT) THEN
          IF(JP(IPP).EQ.KBLANK) THEN
            JP(IPP)=0
          ELSE
            JP(IPP)=1
          ENDIF
        ENDIF
  100 CONTINUE
C
C***OUTPUT JP AND INCREMENT LINE COUNT
C
      WRITE(IU,1110) JP
 1110 FORMAT(A1,27I5)
      IES(IU)=IES(IU)+1
C
      RETURN
      END
C*** OF LDMP
```

```
      SUBROUTINE MFZ (JP,LPL,MP)
C
C PLOTS JP ON MICROFILM. SYMBOLS ARE RECOGNISED AND DRAWN. DATA
C IS REPRESENTED BY SHAPES OF VARYING SIZE, THERE BEING A NUMBER
C OF SHAPES AND PATTERNS AVAILABLE. UP TO THREE RANGES OF DATA,
C PLOTTED AT DIFFERENT INTENSITY, ARE POSSIBLE TO EXTEND THE
C DYNAMIC RANGE. THIS ROUTINE IS CALLED FROM LNPLOT AND MFGS.
C
      INTEGER JP(LPL)
      INTEGER IN(3)
      REAL SC(3),EI(3),CI(3)
      COMMON/STATUS/IEQ,IER,IES(29),MAXDAT,MAXPLT,NGRP
      COMMON/KON/K(128),KONLEV(128),KONSYM(128),KONTAB(2048)
      DATA KBLANK,KA,KB,KC,KD,KE/1H ,1H+,1H=,1H*,1HO,1H-/
      DATA R,JT,JTS,JTA,JTM/1.33057E-3,0,7,3154265510334278,
     +  777777777777777B/
C
C***RETURN ON ERROR OR OVER FRAME LIMIT
      IF(IER.GT.0.OR.IES(11).GT.IES(6)) RETURN
C
C***SET PLOTTING GRID INTERVALS
      IX=16384/(20*MOD(MP,100))
      KSQZ=1+MOD((MP/4000),2)
      IY=IX/KSQZ
      IX=IY*KSQZ
C***CENTRE OF HYPOTHETICAL PRECEDING POINT
      KX=IES(13)
      KY=IES(16)-IY
C***UPDATE X, Y USAGE LIMITS, RETURN IF OUT OF SPACE IN Y
      IES(16)=KY
      IF(IES(16).LT.IY) RETURN
      IES(15)=MAXO(IES(15),(IES(13)+((LPL-1)*IX)))
C***PLOTTING UNIT, OUTPUT LENGTH, THRESHOLD, JITTER SCALE
      NX=IX
      IF(K(48).NE.0) NX=K(48)
      RL=R*FLOAT(NX*K(51))/1024.0
      RT=R*FLOAT(K(52))+RL
      RJT=2.0*R/FLOAT(JTM+1)
C***POSITION CONVERTED TO PLOTTING UNITS (INCHES)
      DX=R*FLOAT(IX)
      CX=R*FLOAT(KX)
      CY=R*FLOAT(KY)
C***HEIGHT AND SLOPE PARAMETERS
      KHT=(NX*K(53))/(1024*KSQZ)
      RHT=R*FLOAT(KHT)
      RSQZ=1.0/FLOAT(KSQZ)
      RS=RSQZ*FLOAT(K(53))/1024.0
C***SET PLOTTING PARAMETERS FOR THE THREE RANGES
      DO 90 I=1,3
        J=49+6*I
        IN(I)=K(J)
        SC(I)=R*FLOAT(NX)/FLOAT(K(J+2))
        EI(I)=R*FLOAT(K(J+3))*0.1
        CI(I)=R*FLOAT(K(J+4))*0.1
   90 CONTINUE
C
C***INVERT FOR LAST DIGIT OF OPTION ODD
      M=MOD(K(47),10)
      IF(MOD(M,2).EQ.1) THEN
        KHT=-KHT
        RHT=-RHT
        RS=-RS
      ENDIF
C***ALTERNATE BETWEEN NORMAL/INVERTED OPTIONS FOR ENDINGS 2/3,6/7
      IF(MOD(M,4).EQ.2) K(47)=K(47)+1
      IF(MOD(M,4).EQ.3) K(47)=K(47)-1
C
C***RETURN IF ONLY A CONTROL CHARACTER. CLEAR DATA FLAG
      IF(LPL.LE.1) RETURN
      IB=0
C
C***LOOP OVER PLOTTING POINTS
C
      DO 100 IPP=2,LPL
C  ***RETURN IF OVER PLOT FILE LENGTH LIMIT
        IF(IES(12).GE.IES(7)) RETURN
C  ***SET POSITION OF NEXT POINT AND RETURN IF OVER LIMITS
        CX=CX+DX
        IF(CX.GT.(21.85-DX)) RETURN
C  ***SAVE CURRENT DATA FLAG, SET NEW VALUE, TEST FOR DATA
        IA=IB
        J=JP(IPP)
        IB=0
        IF(IABS(J).LE.MAXDAT) IB=1
C
C
        IF(IB.EQ.1) THEN
C  ***PLOT A DATA VALUE
C  ***ADD JITTER TO CENTRE POSITION
          JS=JTM.AND.(SHIFT(JT,JTS)+JT+JTA)
          X=CX+RJT*FLOAT(JS)
          JT=JTM.AND.(SHIFT(JS,JTS)+JS+JTA)
          Y=CY+RJT*FLOAT(JT)
C  ***IF NOT PRECEDED BY A DATA POINT, PLOT A DUMMY FIRST
          IF(IA.EQ.0) THEN
            CALL PLOT((X-DX),Y,3)
            IES(12)=IES(12)+1
          ENDIF
C  ***SCALE VALUE, LINEAR OR LOG
          IF(K(44).EQ.0) THEN
            J=(2*K(41)*(J-K(2)))/K(3)
          ELSE
            J=IFIX(FLOAT(K(41))*
     +          ALOG10(AMAX1(1.E-6,(FLOAT(J-K(2))/FLOAT(K(3))))))
          ENDIF
C  ***LIMIT, OPTIONALLY INVERT OR TABLE LOOK UP, FINAL SCALING
          J=MAXO(0,MINO(MAXPLT,J))
          IF(K(46).NE.0) J=MAXPLT-J
          IF(K(45).NE.0) J=KONTAB(J+1)
          J=(J-K(42))/K(43)
C  ***SELECT RANGE
          N=1
          IF(J.GT.K(62)) N=2
          IF(J.GT.K(68)) N=3
C  ***CALCULATE LENGTHS FOR PLOTTING ADDING JITTER
          A=FLOAT(J)*SC(N)+EI(N)+RJT*FLOAT(JT+JS/2)
          B=0.0
          IF(A.GT.RT) THEN
            B=A-RL+CI(N)
            A=RL
          ENDIF
C  ***SET INTENSITY IF DIFFERENT
          IF(IES(8).NE.IN(N)) THEN
            CALL INTENSE(IN(N))
            IES(8)=IN(N)
            IES(12)=IES(12)+1
          ENDIF
C  ***AUX. OPTION AND LENGTH PARAMETERS FOR CONVENIENCE
          M=K(47)
          C=0.5*A
C
C
          IF(M.LE.29) THEN
C  ***OPTION SKEW, STAGGERED, OR DOUBLE
C  ***BASIC OFFSET
            D=RHT
            IF(M.GE.10.AND.MOD(IPP,2).EQ.0) D=-D
C  ***Y LEFT OFFSET
            F=D
            IF(M.LE.5) F=RS*C
            IF(M.EQ.0.OR.M.GE.28) F=0.0
            IF(M.EQ.25.AND.(A+B).GT.0.0) F=D*B/(A+B)
            IF(M.EQ.14.AND.A.LT.(0.8*DX)) F=0.0
C  ***Y RIGHT OFFSET
            H=F
            IF(M.LE.9) H=-F
C  ***X LEFT, RIGHT OFFSETS
            E=-C
            IF(M.EQ.1) E=0.0
            G=-E
C  ***STAGGERED VERTICAL OPTION
            IF(M.EQ.16.OR.M.EQ.17) THEN
              E=RHT
              G=E
              F=RS*C
              H=-F
            ENDIF
C  ***PLOT LINE SEGMENT
            CALL PLOT((X+E),(Y+F),3)
            CALL PLOT((X+G),(Y+H),2)
            IES(12)=IES(12)+2
C  ***DOUBLE LINE OPTION WITH SECOND LINE REQUIRED
            IF(M.GE.20.AND.B.GT.0.0) THEN
              P=0.5*B
              Q=D
              IF(M.LE.23) Q=3*D
              IF(M.EQ.25) Q=D-F
              IF(M.EQ.29) Q=RHT
              CALL PLOT((X-P),(Y-Q),3)
              CALL PLOT((X+P),(Y+Q),2)
              IES(12)=IES(12)+2
            ENDIF
C
```

```
      ELSEIF(M.LE.39) THEN
C       ***OPTION SYMMETRIC Y
        P=C
        Q=RHT
        IF(M.EQ.33) P=-C
        IF(M.EQ.31.AND.MOD(IPP,2).EQ.0) P=-C
        CALL PLOT((X+P),(Y+Q),3)
        CALL PLOT((X-P),(Y  ),2)
        CALL PLOT((X+P),(Y-Q),2)
        IES(12)=IES(12)+3
C
      ELSEIF(M.LE.49) THEN
C       ***OPTION VARIABLE Y
C       ***CENTERING OFFSET
        E=0.0
        IF(M.GE.46.AND.A.GT.0.0.AND.B.GT.0.0) E=0.5*B*(A-B)/(A+B)
C       ***PLOT FIRST SEGMENT
        P=X+C+E
        Q=Y+0.5*RS*(A+B)
        CALL PLOT(P,Q,3)
        P=P-A
        Q=Q-RS*A
        CALL PLOT(P,Q,2)
        IES(12)=IES(12)+2
C       ***SECOND SEGMENT IF NEEDED
        IF(B.GT.0.0) THEN
          P=P+B
          Q=Q-RS*B
          CALL PLOT(P,Q,2)
          IES(12)=IES(12)+1
        ENDIF
      ENDIF
C
C
    ELSEIF(J.NE.KBLANK) THEN
C     ***PLOT A NON BLANK SYMBOL
C     ***CONVERT TO A NUMBER CODING PLOTTING SYMBOL
      KS=9
      IF(J.EQ.KA) KS=3
      IF(J.EQ.KB) KS=6
      IF(J.EQ.KC) KS=4
      IF(J.EQ.KD) KS=11
      IF(J.EQ.KE) KS=13
      KX=IFIX(CX/R)
      CALL MFSYM(KX,KY,(IX/2),(IY/2),0,K(50),-5,KS)
    ENDIF
  100 CONTINUE
C
  RETURN
  END
C*** OF MFZ
```

```
  SUBROUTINE MFC (JP,JQ,LPL,MP)
C
C PLOTS JP ON MICROFILM AS CONTOURS (IN COMPARISON WITH PREVIOUS
C LINE JQ), CALLING MFCL OR MFCSP FOR THE ACTUAL CONTOURING.
C SYMBOLS ARE RECOGNISED AND DRAWN. CURRENT DATA IS COPIED
C INTO JQ. IT REQUIRES THAT ANY SEQUENCE OF LINES OF DATA BE
C PRECEDED BY A PURELY SYMBOLIC LINE. THIS ROUTINE IS CALLED
C FROM LNPLOT AND MFGS.
C
  INTEGER JP(LPL),JQ(LPL)
  COMMON/STATUS/IEQ,IER,IES(29),MAXDAT,MAXPLT,NGRP
  COMMON/KON/K(128),KONLEV(128),KONSYM(128),KONTAB(2048)
  DATA KBLANK,KA,KB,KC,KD,KE/1H ,1H+,1H=,1H*,1HO,1H-/
  DATA KBB,KBT/2H+B,2H+T/
C
C***RETURN ON ERROR OR OVER FRAME LIMIT
  IF(IER.GT.0.OR.IES(11).GT.IES(6)) RETURN
C
C***SET PLOTTING GRID INTERVALS
  KSQZ=1+MOD((MP/4000),2)
  IX=16384/(20*MOD(MP,100))
  IY=IX/KSQZ
  IX=IY*KSQZ
C***CENTRE OF HYPOTHETICAL PRECEDING POINT
  KX=IES(13)
  KY=IES(16)-IY
C***UPDATE X,Y USAGE LIMITS, RETURN IF OUT OF SPACE IN Y
  IES(16)=KY
  IF(IES(16).LT.IY) RETURN
  IES(15)=MAXO(IES(15),(IES(13)+(LPL-1)*IX))
C***RETURN IF ONLY A CONTROL CHARACTER. ZERO DATA POINT COUNTER
  IF(LPL.LE.1) RETURN
  N=0
C***CLEAR BOX COMPLETE FLAG
  JC=0
C
C***LOOP OVER PLOTTING POINTS
C
  DO 100 I=2,LPL
C  ***RETURN IF OVER PLOT FILE LENGTH LIMIT
    IF(IES(12).GE.IES(7)) RETURN
C  ***SET X FOR NEXT POINT. SKIP PROCESSING IF OUTSIDE LIMITS
    KX=KX+IX
    IF((KX+IX).LE.16383) THEN
C     ***TEST FOR DATA VALUE. INCREMENT COUNTER OR CHANGE SIGN
      IF(IABS(JP(I)).LE.MAXDAT) THEN
        N=N+1
      ELSE
        N=-N
      ENDIF
C
C     ***IF A DATA POINT OR ONE IMMEDIATELY FOLLOWING A DATA
C     ***POINT, CALL PLOTTING ROUTINE
      IF(N.NE.0) THEN
        IF(K(29).LE.1) THEN
          IF(N.GT.0) CALL MFCL(JP(I),JQ(I),KX,KY,IX,IY,N)
        ELSE
C         ***HIGHER ORDER INTERPOLATION (UNIMPLEMENTED)
          CALL MFCSP(JP(I),JQ(I),KX,KY,IX,IY,N)
C         ***EXTRA WRAP-UP CALL IF DATA REACHES END OF LINE
          IF(N.GT.0.AND.I.EQ.LPL)
     +       CALL MFCSP(JP(I),JQ(I),(KX+IX),KY,IX,IY,-N)
        ENDIF
      ENDIF
C
C     ***IF NOT A DATA POINT, ZERO COUNT
      IF(N.LT.0) N=0
C
C     ***IF NOT A DATA POINT AND NOT BLANK, CONVERT TO
C     ***NUMBER CODING A SUITABLE SYMBOL AND PLOT
      IF(N.EQ.0) THEN
        IF(JP(I).EQ.KBT) THEN
C         ***TOP OF BOX
          KT=KY
        ELSEIF(JP(I).EQ.KBB) THEN
C         ***BOTTOM OF BOX
          IF(JC.EQ.0) KL=KX
          IF(JC.EQ.1) CALL MBX(KL+IX,KX-IX,KY+IY,KT-IY,K(33))
          JC=1
        ELSEIF(JP(I).NE.KBLANK) THEN
          M=9
          IF(JP(I).EQ.KA) M=3
          IF(JP(I).EQ.KB) M=6
          IF(JP(I).EQ.KC) M=4
          IF(JP(I).EQ.KD) M=11
          IF(JP(I).EQ.KE) M=13
          CALL MFSYM(KX,KY,(IX/2),(IY/2),0,K(33),-5,M)
        ENDIF
      ENDIF
    ENDIF
C  ***COPY JP TO JQ
    JQ(I)=JP(I)
  100 CONTINUE
C
  RETURN
  END
C*** OF MFC
```

```
      SUBROUTINE MFCL (IP,IQ,XX,XY,IX,IY,N)
C
C CONTOURS ON MICROFILM ONE CELL OF DATA, DIMENSIONS IX,IY,
C LOWER RIGHT CORNER AT XX,XY, LOWER RIGHT DATA VALUE IS IP,
C UPPER RIGHT IQ. THE LEFT VALUES ARE ASSUMED SUPPLIED BY THE
C PREVIOUS CALL. THE COUNT N OF SUCCESSIVE DATA VALUES IN THE
C CURRENT (LOWER) LINE ENABLES THE FIRST CALL, WITH NO
C PRECEDING DATA, TO BE RECOGNISED. THIS ROUTINE IS CALLED
C FROM MFC.
C
      INTEGER J(2),L(2)
      REAL XP(4),YP(4)
      COMMON/STATUS/IEQ,IER,IES(29),MAXDAT,MAXPLT,NGRP
      COMMON/KON/K(128),KONLEV(128),KONSYM(128),KONTAB(2048)
C
C***IF FIRST DATA POINT OR NOT DATA, UPDATE OLD VALUES, RETURN
      IF(N.LE.1) THEN
        JP=IP
        JQ=IQ
        RETURN
      ENDIF
C***IF CURRENT OR OLD VALUE ON PREVIOUS LINE NOT DATA, RETURN
      IF(IABS(IQ).GT.MAXDAT.OR.IABS(JQ).GT.MAXDAT) RETURN
C
C***MAX AND MIN VALUES OF THE 4 CORNER DATA VALUES
      J(1)=MAXO(JP,JQ,IP,IQ)
      J(2)=MINO(JP,JQ,IP,IQ)
C***CONVERT TO LEVELS
      DO 100 I=1,2
        IF(K(26).EQ.0) THEN
          L(I)=(2*K(23)*(J(I)-K(2)))/K(3)
        ELSE
          L(I)=IFIX(FLOAT(K(23))*ALOG10(AMAX1
     +          (1.E-6,(FLOAT(J(I)-K(2))/FLOAT(K(3))))))
        ENDIF
        L(I)=MAXO(0,MINO(MAXPLT,L(I)))
        IF(K(27).NE.0) L(I)=KONTAB(L(I)+1)
        LTMP=((L(I)-K(24))/NGRP)/K(25)
        L(I)=MINO(100,MAXO(0,LTMP))
  100 CONTINUE
C***NUMBER OF LEVELS CROSSED. IF ZERO SKIP PROCESSING
      NLEV=IABS(L(1)-L(2))
      IF(NLEV.NE.0) THEN
C
C  ***POSITION IN PLOTTING UNITS
        R=1.33057E-3
        X=R*FLOAT(KX)
        Y=R*FLOAT(KY)
C  ***IF AN EXCESSIVE NUMBER OF CROSSINGS, SET A
C  ***SELECTION FACTOR GREATER THAN 1
        IF(NLEV.GT.K(32)) THEN
          ID=1+(NLEV-1)/K(32)
          IF(ID.GE.K(28)) THEN
            ID=(((ID-1)/K(28))+1)*K(28)
          ELSEIF(ID.GT.4) THEN
            ID=(((ID-1)/8)+1)*8
          ELSEIF(ID.GT.2) THEN
            ID=4
          ENDIF
        ELSE
          ID=1
        ENDIF
C  ***UPPER AND LOWER LEVELS
        LL=MINO(L(1),L(2))
        LH=MAXO(L(1),L(2))
C
C  ***LOOP OVER CROSSINGS OF LEVELS
C
        DO 120 LEV=(LL+1),LH
C    ***SKIP PROCESSING ACCORDING TO SELECTION FACTOR
          IF(MOD(LEV,ID).EQ.0) THEN
C      ***LEVEL VALUE
            JLEV=(LEV*K(25)*NGRP)+K(24)
            IF(K(27).NE.0) JLEV=KONLEV(JLEV+1)
            IF(K(26).EQ.0) THEN
              JLEV=(JLEV*IABS(K(3))-1)/(2*IABS(K(23)))+1
              IF(K(3)*K(23).LT.0) JLEV=-JLEV
              JLEV=JLEV+K(2)
            ELSE
              JLEV=IFIX(0.999999999+(10.0**(FLOAT(JLEV)/
     +          FLOAT(K(23))))*FLOAT(K(3)))+K(2)
            ENDIF
C      ***SELECT AND SET INTENSITY ACCORDING TO LEVEL
C      ***NUMBER AND EMPHASIS PERIOD
            IT=K(35)
            IF(MOD(LEV,K(28)).NE.0) IT=K(34)
            IF(IES(8).NE.IT) THEN
              CALL INTENSE(IT)
              IES(8)=IT
              IES(12)=IES(12)+1
            ENDIF
C
C      ***LOOP OVER EDGES OF CELL
            DO 110 I=1,4
C        ***SET END VALUES, LENGTH AND OFFSET FOR EACH SIDE
              IF(I.EQ.1) THEN
                JA=IP
                JB=JP
                LEN=-IX
                IOF=0
C          ***ZERO POINTS PLOTTED COUNTER
                NC=0
              ELSEIF(I.EQ.2) THEN
                JB=IQ
                LEN=IY
              ELSEIF(I.EQ.3) THEN
                JA=IQ
                JB=JQ
                LEN=-IX
                IOF=IY
              ELSE
                JA=JP
                LEN=IY
                IOF=-IX
              ENDIF
C        ***IF LEVEL VALUE LIES OUTSIDE END VALUES SKIP PROC.
              IF((JLEV.LE.JA.AND.JLEV.GT.JB).OR.
     +           (JLEV.LE.JB.AND.JLEV.GT.JA)) THEN
C          ***POSITION OF CROSSING, OFFSET IN PLOTTING UNITS
                DEL=R*FLOAT((LEN*(JLEV-JA))/(JB-JA))
                DOFF=R*FLOAT(IOF)
C          ***INCREMENT COUNT OF PLOTTING POINTS
                NC=NC+1
C          ***SAVE COORDINATES OF INTERSECTION
                IF(I.EQ.1.OR.I.EQ.3) THEN
                  XP(NC)=X+DEL
                  YP(NC)=Y+DOFF
                ELSE
                  XP(NC)=X+DOFF
                  YP(NC)=Y+DEL
                ENDIF
              ENDIF
  110       CONTINUE
C
C      ***PLOT LINE SEGMENTS
C
            JCON=0
            IF(NC.EQ.4) THEN
C        ***SELECT PAIRS TO BE CONNECTED
              SA=ABS(XP(1)-XP(2))+ABS(YP(1)-YP(2))
              SB=ABS(XP(2)-XP(3))+ABS(YP(2)-YP(3))
              SC=ABS(XP(3)-XP(4))+ABS(YP(3)-YP(4))
              SD=ABS(XP(4)-XP(1))+ABS(YP(4)-YP(1))
              IF((SB+SD).LT.(SA+SC)) JCON=1
            ENDIF
C
            IF(NC.GT.1) THEN
              KP=3
              DO 115 IC=1,NC
                NPT=IC+JCON
                IF(NPT.GT.4) NPT=1
                CALL PLOT(XP(NPT),YP(NPT),KP)
                IES(12)=IES(12)+1
                KP=2
                IF(NC.EQ.4) KP=2+MOD(IC+1,2)
  115         CONTINUE
            ENDIF
          ENDIF
  120   CONTINUE
      ENDIF
C
C***UPDATE PREVIOUS INPUT VALUES
      JP=IP
      JQ=IQ
C
      RETURN
      END
C*** OF MFCL
```

```
      SUBROUTINE MFPAGE (MF,MO,MP,LPL)
C
C SETS UP A NEW FRAME ON MICROFILM. IF THE FIRST FRAME IT MAKES
C THE CALL TO START, AND IF THE LAST HAS BEEN REACHED IT TAKES
C NO ACTION. IF IT IS CALLED AFTER THE PLOT HAS BEEN CLOSED
C (MARKED BY A NEGATIVE FRAME COUNT) IT IS TREATED AS AN ERROR.
C THE ORIGIN POSITION IS SET IN IES. IF REQUIRED BY MF A CALL
C TO MFGS SETS UP A BORDER OR A GREY SCALE. THIS ROUTINE IS
C CALLED FROM PRORDY, MFZ, AND MFC.
C
      COMMON/STATUS/IEQ,IER,IES(32)
C
      IF(IES(11).EQ.0) THEN
C  ***FIRST FRAME
        CALL START(2)
        CALL BORDOFF
        CALL NEWPAGE
      ELSEIF(IES(11).GT.0.AND.IES(11).LT.IES(6)) THEN
C  ***NORMAL
        CALL NEWPAGE
      ELSEIF(IES(11).GE.IES(6)) THEN
C  ***FRAME LIMIT REACHED
        IES(11)=IES(11)+1
        RETURN
      ELSE
C  ***PLOT CLOSED
        IER=6001
        IF(IEQ.GT.0) CALL ERROR
        RETURN
      ENDIF
C
C***CANCEL INTENSITY SETTING
      IES(8)=0
C
C***INCREASE FRAME COUNT AND APPROXIMATE PLOT FILE LENGTH
      IES(11)=IES(11)+1
      IES(12)=IES(12)+2
C***SET ORIGINS ACCORDING TO FRAME SIZE SELECTED BY MP
      M=MOD((MP/1000),4)
      IES(13)=0
      IF(M.EQ.2) IES(13)=1365
      IES(14)=12288
      IF(M.EQ.1) IES(14)=10923
      IF(M.EQ.2) IES(14)=10012
      IF(M.EQ.0.OR.M.EQ.3) CALL PLOT(-1.4,-0.92,-3)
      IF(M.EQ.1) CALL PLOT(-1.4,-0.02,-3)
      IF(M.EQ.2) CALL PLOT(-1.4, 0.59,-3)
C***SET POSITION FOR GREY SCALE AND CALL MFGS
      IES(15)=13880
      IES(16)=IES(14)
      IF(MOD((MP/100),10).GT.0) CALL MFGS(MO,MP)
C***SET ORIGIN TO APPROXIMATELY CENTRE PLOT (UNLESS MERGED)
      IF(MOD((MF/100),10).NE.1) THEN
        L=(LPL-1)*(16384/(20*MOD(MP,100)))
        IES(13)=IES(13)+MAXO(0,(5850-L/2))
      ENDIF
C***RESET CURRENT POSITION
      IES(15)=IES(13)
      IES(16)=IES(14)
C
      RETURN
      END
C*** OF MFPAGE
C
C
C
      SUBROUTINE MFGS (MO,MP)
C
C GENERATES A STEPPED OR CONTINUOUS SCALE OF VALUES AND CAUSES
C IT TO BE OUTPUT TO MICROFILM, WITH OR WITHOUT NUMERIC LABELS.
C THE TYPE IS GIVEN BY MP. IT CAN ALSO DRAW A PARTIAL BORDER
C ROUND THE ENDS OF THE FRAME. THIS ROUTINE IS CALLED FROM
C MFPAGE.
C
      INTEGER JP(16),JQ(16)
      COMMON/STATUS/IEQ,IER,IES(29),MAXDAT,MAXPLT,NGRP
      COMMON/KON/K(128),KONLEV(128),KONSYM(128),KONTAB(2048)
      DATA KBLANK,JOLD/1H ,0/
C
C***BORDER
C
      IF(MOD((MP/100),2).EQ.1) THEN
        CALL MFSYM((IES(13)+450),IES(14),0,0,0,20,-7,0)
        CALL MFSYM( IES(13)     ,IES(14),0,0,0,20,-8,0)
        CALL MFSYM( IES(13)     , 0     ,0,0,0,20,-8,0)
        CALL MFSYM((IES(13)+450), 0     ,0,0,0,20,-8,0)
        CALL MFSYM((16383 - 450),IES(14),0,0,0,20,-7,0)
        CALL MFSYM( 16383       ,IES(14),0,0,0,20,-8,0)
        CALL MFSYM( 16383       , 0     ,0,0,0,20,-8,0)
        CALL MFSYM((16383 - 450), 0     ,0,0,0,20,-8,0)
        IES(13)=IES(13)+600
      ENDIF
C
C***ISOLATE TYPE OF SCALE, AND RETURN IF NOT REQUIRED
      MODE=MOD((MP/200),5)
      IF(MODE.EQ.0) RETURN
C***SAVE X ORIGIN AND RESET TO POSITION PROVIDED
      KXHOLD=IES(13)
      IES(13)=IES(15)
C***FIND PLOTTING GRID INTERVALS
      IX=16384/(20*MOD(MP,100))
      KSQZ=1+MOD((MP/4000),2)
      IY=IX/KSQZ
      IX=IY*KSQZ
C***FIND LINES TO A STEP, NUMBER OF STEPS, POINTS TO A LINE
      NL=(IES(16)-IY-3000)/(11*IY)
      NL=MIN0(10,MAX0(4,NL))
      NS=(IES(16)-IY-3000)/(NL*IY)
      NS=MIN0(11,MAX0(2,NS))
      NP=1500/IX
      NP=MIN0(15,NP,(2*NL))
C***SET FLAGS FOR STEPPED AND FOR CONTOURING
      MS=0
      IF(MODE.EQ.1.OR.MODE.EQ.3) MS=1
      MC=0
      IF((MO/100).EQ.6) MC=1
C***SELECT SCALING CONSTANTS AND LOG MODE FLAG
      KEXP=K(41)
      IF(MC.EQ.1) KEXP=K(23)
      KLOG=K(44)
      IF(MC.EQ.1) KLOG=K(26)
C***SET FLAG FOR ALTERNATING VALUES BETWEEN LINES (ONE CONT MODE)
      MA=0
      IF(MC.EQ.1.AND.(K(29).EQ.2.OR.
     +           (K(29).EQ.0.AND.IABS(K(30)).EQ.1))) MA=1
C***PLOT A BLANK LINE (NEEDED BY MFC)
      DO 100 I=1,16
        JP(I)=KBLANK
  100 CONTINUE
      IF(MC.EQ.0) CALL MFZ(JP,(NP+1),MP)
      IF(MC.EQ.1) CALL MFC(JP,JQ,(NP+1),MP)
C
C***LOOP OVER STEPS
C
      DO 130 IS=1,NS
C  ***IF OUT OF TIME SKIP PROCESSING IN LOOP
        IF(SECOND(X).LT.FLOAT(IES(5))) THEN
C    ***VALUE FOR BOTTOM LINE OF STEP, RANGE 0...MAXPLT
          JS=(MAXPLT*(NS-IS))/(NS-1)
C    ***LOOP OVER LINES OF STEP
          DO 120 IL=1,NL
C      ***VALUE FOR LINE
            J=JS
            IF(MS.EQ.0) J=(MAXPLT*(NL*(NS-IS)+(NL-IL)))/(NL*(NS-1))
C      ***SKIP PROCESSING OF LINE IF CONTINUOUS SCALE AND
C      ***BEFORE LAST LINE OF FIRST STEP
            IF(MS.EQ.1.OR.IS.GT.1.OR.IL.EQ.NL) THEN
C        ***CONVERT TO ACTUAL DATA VALUE
              IF(KLOG.EQ.0) THEN
                JO=K(2)+((J*K(3))/(2*KEXP))
              ELSE
                JO=K(2)+IFIX(0.5+FLOAT(K(3))*
     +                          (10.0**(FLOAT(J)/FLOAT(KEXP))))
              ENDIF
C        ***COPY VALUE INTO PLOTTING ARRAY
              DO 110 I=2,16
                JP(I)=JO
                IF(MA.EQ.1) THEN
C                 ***ALTERNATE WITH VALUE FROM PREVIOUS LINE
                  IF(MOD(I,2).NE.0.AND.(IS.GT.1.OR.IL.GT.1)) JP(I)=JOLD
                ENDIF
  110         CONTINUE
C        ***PLOT LINE AND SAVE DATA VALUE
              IF(MC.EQ.0) CALL MFZ(JP,(NP+1),MP)
              IF(MC.EQ.1) CALL MFC(JP,JQ,(NP+1),MP)
              JOLD=JO
            ENDIF
  120     CONTINUE
C    ***PLOT LABEL AT END OF STEP IF REQUIRED
          IF(MODE.GE.3) THEN
            IF(KLOG.EQ.0) THEN
              JCAP=K(2)+(NS-IS)*(((MAXPLT+1)*K(3))/(2*KEXP*(NS-1)))
              NCAP=0
            ELSE
              JCAP=(NS-IS)*(((MAXPLT+1)*1000)/(KEXP*(NS-1)))
              NCAP=-3
            ENDIF
            CALL MFSYM((IES(15)+220),IES(16),30,30,0,20,-5,3)
            CALL MFSYM((IES(15)+330),IES(16),30,30,0,20,-5,3)
            CALL MFSYM((IES(15)+440),IES(16),110,110,0,20,NCAP,JCAP)
          ENDIF
        ENDIF
  130 CONTINUE
C
C***ADD LOG LABEL IF NEEDED
      IF(MODE.GE.3.AND.KLOG.NE.0) THEN
        CALL MFSYM((IES(15)+440),IES(16)-480,160,160,0,20,3,3HLOG)
      ENDIF
C
C***RESTORE X ORIGIN
      IES(13)=KXHOLD
C
      RETURN
      END
C*** OF MFGS
```

```
 SUBROUTINE MFEND (ISEC,NSEC,IBLK,NBLK,KIDA,KIDB,K)
C
C PRODUCES A LABEL ON MICROFILM EITHER BELOW THE PLOT OR IN THE
C LOWER RIGHT CORNER ACCORDING TO THE PARAMETER K. THE OTHER
C PARAMETERS PROVIDE THE TEXT. THIS ROUTINE IS CALLED FROM
C BLKEND.
C
 COMMON/STATUS/IEQ,IER,IES(32)
C
C***RETURN IF OVER FRAME OR PLOT FILE LIMITS
 IF(IES(11).GT.IES(6).OR.IES(12).GT.IES(7)) RETURN
C
 IF(K.GE.5) THEN
C  ***LABEL UNDER BLOCK, AT SIDE IF INSUFFICIENT ROOM
   IF(IES(16).GE.490) THEN
     IES(16)=IES(16)-490
     CALL MFSYM( IES(13)     ,IES(16),110,110,0,20,0,KIDA)
     CALL MFSYM((IES(13)+660),IES(16),110,110,0,20,0,KIDB)
   ELSE
     CALL MFSYM((IES(15)+330),  220  ,110,110,0,20,0,KIDA)
     CALL MFSYM((IES(15)+330),    0  ,110,110,0,20,0,KIDB)
   ENDIF
 ELSE
C  ***LABEL IN CORNER
     CALL MFSYM(13560,440,160,160,0,20,4,4HPLOT)
     CALL MFSYM(14360,440,160,160,0,20,0,KIDA)
     CALL MFSYM(15160,440,160,160,0,20,0,KIDB)
   IF(K.EQ.2.OR.K.EQ.4) THEN
     CALL MFSYM(13880,220,110,110,0,20,3,3HSEC)
     CALL MFSYM(14320,220,110,110,0,20,0,ISEC)
     CALL MFSYM(14760,220,110,110,0,20,2,2HOF)
     CALL MFSYM(15090,220,110,110,0,20,0,NSEC)
   ENDIF
   IF(K.EQ.3.OR.K.EQ.4) THEN
     CALL MFSYM(13880,  0 ,110,110,0,20,3,3HBLK)
     CALL MFSYM(14320,  0 ,110,110,0,20,0,IBLK)
     CALL MFSYM(14760,  0 ,110,110,0,20,2,2HOF)
     CALL MFSYM(15090,  0 ,110,110,0,20,0,NBLK)
   ENDIF
 ENDIF
C
 RETURN
 END
C*** OF MFEND
```

```
 SUBROUTINE MFSYM (KX,KY,IX,IY,IA,IT,M,N)
C
C PLOTS ON MICROFILM CHARACTER STRINGS, NUMBERS, SYMBOLS, LINES
C AND POINTS ACCORDING TO MODE M AND VALUE N, AT INTENSITY IT.
C POSITION ANGLE (FOR TEXT) IA, AND POSITION KX, KY. CHARACTER
C SIZE IS SPECIFIED BY IX, IY. THIS ROUTINE IS USED BY MOST OF
C THE MICROFILM OUTPUT ROUTINES.
C
 COMMON/STATUS/IEQ,IER,IES(32)
C
C***SET UP REAL PARAMETERS USED BY MICROFILM ROUTINES
 R=1.33057E-3
 X=R*FLOAT(KX)
 Y=R*FLOAT(KY)
 H=R*FLOAT(MINO(IX,IY))
 A=FLOAT(IA)
C
C***SET INTENSITY IF DIFFERENT FROM CURRENT VALUE. 0=UNALTERED
 IF(IT.GT.0.AND.IT.NE.IES(8)) THEN
   CALL INTENSE(IT)
   IES(8)=IT
   IES(12)=IES(12)+1
 ENDIF
C
C
 IF(M.GT.0) THEN
C  ***OUTPUT M CHARACTERS FROM SOURCE N (CAN BE AN ARRAY NAME)
   CALL SYMBOL(X,Y,H,N,A,M)
   NC=8*M
 ELSEIF(M.EQ.0) THEN
C  ***OUTPUT INTEGER N
   CALL NUMBER(X,Y,H,FLOAT(N),A,-1)
   NC=30
 ELSEIF(M.GE.-4) THEN
C  ***OUTPUT INTEGER N AS FP NUMBER N*(10**M)
   FP=FLOAT(N)
   DO 100 I=1,-M
     FP=FP/10.0
   100 CONTINUE
   CALL NUMBER(X,Y,H,FP,A,-M)
   NC=40
 ELSEIF(M.GE.-6) THEN
C  ***SINGLE SYMBOL NUMBER N, M=-5. WITH LINE TO SYMBOL M=-6
   CALL SYMBOL(X,Y,H,N,A,(M+4))
   NC=8
 ELSEIF(M.GE.-8) THEN
C  ***LINE, M=-8, OR POSITION WITHOUT WRITING, M=-7
   CALL PLOT(X,Y,(M+10))
   NC=1
 ELSEIF(M.EQ.-9) THEN
C  ***POINT
   CALL POINT(X,Y)
   NC=1
 ELSE
   NC=0
 ENDIF
C
C***INCREASE PLOT FILE LENGTH BY APPROXIMATE COUNT NC
 IES(12)=IES(12)+NC
C
 RETURN
 END
C*** OF MFSYM
C
C
C
 SUBROUTINE MBX (KL,KR,KB,KT,IT)
C
C PLOTS A RECTANGLE ON MICROFILM, INTENSITY IT, DIAGONAL
C CORNERS KL,KB KR,KT.
C
 COMMON/STATUS/IEQ,IER,IES(32)
C
 R=1.33057E-3
 XL=R*KL
 XR=R*KR
 YB=R*KB
 YT=R*KT
C
C***SET INTENSITY IF CHANGED. ZERO = UNALTERED
 IF(IT.GT.0.AND.IT.NE.IES(8)) THEN
   CALL INTENSE(IT)
   IES(8)=IT
   IES(12)=IES(12)+1
 ENDIF
C
 CALL PLOT(XL,YB,3)
 CALL PLOT(XL,YT,2)
 CALL PLOT(XR,YT,2)
 CALL PLOT(XR,YB,2)
 CALL PLOT(XL,YB,2)
C
C***INCREASE PLOT FILE LENGTH
 IES(12)=IES(12)+5
C
 RETURN
 END
C*** OF MBX
```

A2
Programs:
Image Generation

A2.1 Star Table Elements

The elements of each entry have a meaning depending on what is represented, except for the first two.

1, 2 :x,y position coordinates
Stars
3 :brightness, peak
Double Stars
3, 4 :brightness (total, peak), position angle
7, 8 :ratio, separation
When converted to stars, 1, 2 and 5, 6 are the two positions; 3, 7 the two brightnesses; and 4, 8 retain the position angle and separation.
Elliptical Galaxies
3, 4 :position angle, radius
5, 6 :ellipticity, integrated brightness
7–12 are used internally.

A2.2 Parameters Required

Some of the main options only are noted.

STRGEN

spread :star image sigma, relative to the line length=1. Point stars if 0. Centred on grid point if negative. For galaxies, use 1 or -1.

cutoff :the level at which the outer parts of images are cut off, 0 giving an acceptable default.

mode :0 for star images (gaussian), 4 for elliptical galaxies.

NOISE

mode :0 for additive noise, 2 for noise proportional to the square root of the signal.

a, b :amplitude of noise, and offset added to signal in mode 2 (otherwise ignored).

seed :initialising value for the pseudorandom noise, 0.0–1.0.

FLTXY

lflt :width of filter in pixels.

sflt :gaussian filter sigma in pixels. If 0, filter is uniformly weighted.

Subsequent parameters can be zero in normal use.

GRDSET

ip :selects the star table element to be modified.

md :type of modification. If the following parameters are p, q, r:

- 0, 1, 2 — add p (or set to zero if p zero) to all , row q, or column q.
- 3 — add random number on interval p–q, with seed r (not reinitialised if r=0).
- 4 — antilog, multiplying by p if not zero.
- 5 — scaling, multiply by p and add q.
- 10, 11, 12 — add arithmetic series from p increasing by q every r (0 defaults to 1) elements.
- 20, 21, 22 — the same but a geometric series.

A2.3 Routines

Routines are listed by name only, and then with brief descriptive notes. The source code follows.

```
SUBROUTINE STRGEN (IA,IB,ST,LST,NSTEL,D,LD,SPREAD,CUTOFF,MD)
SUBROUTINE ICOMPF (ST,LST,NSTEL,IST,GRID,X,Y,OUT)
SUBROUTINE INITCF (ST,LST,NSTEL,GRID,CUT,OUT)

SUBROUTINE GRDCLR (S,LS,NSTEL,NS,NR,NC,NROW,NCOL,SEED)
SUBROUTINE GRDDEF (S,LS,NSTEL,NR,NC,XL,XH,YL,YH,AMP)
SUBROUTINE GRDDBS (S,LS,NSTEL,NR,NC,JCON)
SUBROUTINE GRDSET (S,LS,NSTEL,NR,NC,IP,MD,X,Y,Z)
SUBROUTINE GRDRND (S,LS,NSTEL,NR,NC,IP)

SUBROUTINE STRCLR (NS)
SUBROUTINE STRSET (ST,LST,NSTEL,NS,R,LR)
SUBROUTINE STRMOD (ST,LST,NSTEL,R,LR)
SUBROUTINE STRLST (IU,ST,LST,NSTEL,NS)
SUBROUTINE STRFLD (NS,GAP,BGAP,SEED,SHAPE)
SUBROUTINE STRTST (ST,LST,NSTEL,NS,NTEST,JGAP)
SUBROUTINE STRINS (ST,LST,NSTEL,NS,BRIGHT,NUM)
SUBROUTINE STRDBL (ST,LST,NSTEL,NS,FAINT,RATIO,SEP,NUM)

SUBROUTINE NOISE  (IA,IB,MODE,D,LD,A,B,SEED)
SUBROUTINE RDG    (Z,N)
SUBROUTINE FLT    (R,LR,IA,IFLT,IB)
SUBROUTINE FLTXY  (R,LR,IA,IB,LFLT,SFLT,MFLT,LVERT,SVERT,MVERT)

SUBROUTINE PKFIND (R,LR,IA,IDEL,NMAX)
SUBROUTINE PKINIT (NMX,IDEL,L,N,IA)
SUBROUTINE PKPRO  (RMAX,IMAX,IL)
SUBROUTINE PKIN   (RMAX,IMAX,IL)
SUBROUTINE PKDEL  (I)
SUBROUTINE PKSWP  (IL)
SUBROUTINE PKLST  (IOUT,IBOR)
SUBROUTINE PKCEN  (R,LR,IA,IB,IBOR)
SUBROUTINE PEAKS  (R,LR,IA,IDEL,NMAX,IBOR,IB,IC)

SUBROUTINE FLSUM  (R,LR,S,LS,IA,IB,IC,SCALEA,SCALEB,ADDC,ITRIM)
SUBROUTINE FLLST  (IA,IB,MPTS,MLINES,SCALE)
SUBROUTINE FLSTAT (R,LR,IA,BASE,RANGE,IUB,IC,BLOG)
SUBROUTINE FLDATA (IA,IU,IATYPE,LL,NL)
SUBROUTINE FLSET  (IA,IB,INCL,INCN)
SUBROUTINE FLSIZE (IA,LL,NL)
SUBROUTINE GETLN  (IA,DATA,LD,TEXT,LT,J,K)
SUBROUTINE PUTLN  (IA,DATA,LD,TEXT,LT,J,K)
```

```
SUBROUTINE STRGEN (IA,IB,ST,LST,NSTEL,D,LD,SPREAD,CUTOFF,MD)
  COPIES FROM FILE IA TO IB, ADDING STAR IMAGES AS SPECIFIED IN
  IN ARRAY ST(LST) WITH NSTEL ELEMENTS PER STAR, OF A PROFILE
  SELECTED BY MD AND SCALES TO A WIDTH SET BY SPREAD. A
  THRESHOLD LEVEL FOR OUTER PARTS IS SUPPLIED AS CUTOFF,
  ZERO DEFAULTING TO 0.0001. ARRAY D(LD) IS USED TO STORE
  THE LINE BEING PROCESSED. MD 4 OR 5 SELECTS EXTERNALLY
  GENERATED IMAGE COMPONENTS.

SUBROUTINE ICOMPF (ST,LST,NSTEL,IST,GRID,X,Y,OUT)
  RETURNS AN INTENSITY POINT ON AN ELLIPTICAL GALAXY IMAGE,
  SELECTED FROM TABLE ST(LST) BY IST, AT POSITION X,Y.
  FOR IST NEGATIVE IT RETURNS THE MAXIMUM RADIUS OF THE
  GALAXY -IST. FOR IST ZERO IT PERFORMS INITIALISING
  CONVERSIONS ON TABLE AND RETURNS THE OVERALL MAXIMUM
  RADIUS, GIVEN FRACTION OF FLUX THAT CAN BE CUT OFF AS Y.

SUBROUTINE INITCF (ST,LST,NSTEL,GRID,CUT,OUT)
  INITIALISES TABLE ST(LST) OF ELLIPTICAL GALAXIES.

SUBROUTINE GRDCLR (S,LS,NSTEL,NS,NR,NC,NROW,NCOL,SEED)
  CLEARS TABLE S(LS), SETS NUMBER OF ROWS AND COLUMNS TO
  NROW,NCOL AND NS TO TOTAL NUMBER OF ELEMENTS. RANDOM
  NUMBER GENERATOR IS INITIALISED UNLESS SEED IS NEGATIVE.

SUBROUTINE GRDDEF (S,LS,NSTEL,NR,NC,XL,XH,YL,YH,AMP)
  ENTERS A RECTANGULAR GRID OF COORDINATES IN TABLE S(LS),
  X FROM XL TO XH, Y FROM YL TO YH, AND ADDS RANDOM VALUES
  IN THE RANGE -AMP TO AMP TO EACH COORDINATE.

SUBROUTINE GRDDBS (S,LS,NSTEL,NR,NC,JCON)
  CONVERTS TABLE S(LS) FROM DOUBLE STARS TO INDIVIDUAL STARS,
  EACH PAIR OF THE GIVEN BRIGHTNESS AND THIS MULTIPLIED BY THE
  GIVEN RATIO. IF FLAG JCON IS POSITIVE, THE TOTAL BRIGHTNESS
  IS MADE EQUAL TO THE GIVEN VALUE.

SUBROUTINE GRDSET (S,LS,NSTEL,NR,NC,IP,MD,X,Y,Z)
  SETS VALUES OF ELEMENT IP (OUT OF NSTEL ELEMENTS) IN TABLE
  S(LS) ACCORDING TO MODE MD AND THREE INPUT PARAMETERS.

SUBROUTINE GRDRND (S,LS,NSTEL,NR,NC,IP)
  RANDOMISES THE ORDER OF ELEMENTS IN POSITION IP IN TABLE S(LS).

SUBROUTINE STRCLR (NS)
  CLEARS THE TABLE OF STARS BY RESETTING THE COUNT OF
  ELEMENTS TO ZERO.

SUBROUTINE STRSET (ST,LST,NSTEL,NS,R,LR)
  ENTERS NSTEL VALUES TAKEN FROM R(LR) INTO THE NEXT
  PLACES IN ST(LST) AND ADVANCES NS TO LAST ELEMENT.

SUBROUTINE STRMOD (ST,LST,NSTEL,R,LR)
  ALTERS AN ENTRY IN ST(LST) SELECTED BY THE FIRST
  DATA VALUE TO CONTAIN THE FOLLOWING DATA VALUES.
```

```
SUBROUTINE STRLST (IU,ST,LST,NSTEL,NS)
  LISTS THE TABLE ST(LST) UP TO ELEMENT NS IN LINES
  OF NSTEL VALUES, LINES NUMBERED.

SUBROUTINE STRFLD (NS,GAP,BGAP,SEED,SHAPE)
  CLEARS STAR TABLE, SETS GAP REQUIRED BETWEEN STARS
  IN /STRCOM/ AND BORDER GAP, AND SHAPE OF FIELD (0 DEFAULTS
  TO 1), AND INITIALISES RANDOM NUMBER GENERATOR.

SUBROUTINE STRTST (ST,LST,NSTEL,NS,NTEST,JGAP)
  TESTS LAST NTEST ENTRIES IN ST(LST) OUT OF THE ENTRIES
  EACH OF NSTEL ELEMENTS (TOTAL NS ELEMENTS) FOR DISTANCE
  FROM ALL PRECEDING ENTRIES AND FROM EDGE. JGAP IS SET TO 1
  IF SPACING IS AT LEAST STRGAP,BORGAP RESPECTIVELY, OTHERWISE 0.

SUBROUTINE STRINS (ST,LST,NSTEL,NS,BRIGHT,NUM)
  ADDS NUM (0 DEFAULTS TO 1) STARS OF BRIGHTNESS BRIGHT TO
  STAR TABLE ST(LST) (ENTRIES OF NSTEL ELEMENTS, TOTAL NS)
  IN RANDOM POSITIONS BUT REQUIRING A GAP STRGAP BETWEEN
  STARS AND BORGAP AT EDGE. AFTER 100 FAILURES THE ATTEMPT
  WILL BE ABANDONED.

SUBROUTINE STRDBL (ST,LST,NSTEL,NS,FAINT,RATIO,SEP,NUM)
  ADDS NUM (0 DEFAULTS TO 1) DOUBLE STARS TO TABLE ST(LST)
  (ENTRIES OF NSTEL ELEMENTS, TOTAL NS). FAINT GIVES THE
  FAINTER COMPONENT, THE OTHER BEING GREATER BY A FACTOR
  RATIO. SEP IS GIVEN BUT PA IS RANDOM, AND IS WRITTEN TO
  THE 4TH ELEMENT OF THE FIRST ENTRY IF THERE IS ROOM (AS
  POSITION ANGLE OF WEAKER COMPONENT ANTICLOCKWISE FROM
  X AXIS). AFTER 100 FAILURES THE ATTEMPT WILL BE ABANDONED.

SUBROUTINE NOISE (IA,IB,MODE,D,LD,A,B,SEED)
  ADDS RANDOM NOISE TO FILE IA, COPYING TO IB. TYPE OF NOISE
  IS SPECIFIED BY MODE, WITH PARAMETERS A AND B, DERIVED FROM
  INITIAL VALUE SEED. ARRAY D(LD) IS TEMPORARY STORAGE FOR
  LINE.

SUBROUTINE RDG (Z,N)
  RANDOM DEVIATE GENERATOR. RETURNS VALUE Z, NORMAL DEVIATE
  FOR N=0, UNIFORM FOR N=-1, RANDOM SIGN FOR N=-2. N GREATER
  THAN 0 CAUSES INITIALISATION TO A POINT IN SEQUENCE
  SELECTED BY N AND BY Z. Z IS IN RANGE 0.0 ... 1.0.

SUBROUTINE FLT (R,LR,IA,IFLT,IB)
  FILTERS ARRAY OF DATA FROM IA ONTO IB BY CONVOLVING
  WITH THE ARRAY FROM FILE IFLT. R(LR) IS THE ARRAY USED
  TO STORE THE FILTER AND THE LINES BEING PROCESSED.
```

```
SUBROUTINE FLTXY (R,LR,IA,IB,LFLT,SFLT,MFLT,LVERT,SVERT,MVERT)
  FILTERS ARRAY OF DATA FROM IA ONTO IB, APPLYING SEPARATE
  X AND Y COMPONENTS. EACH IS SPECIFIED BY A LENGTH IN TERMS
  OF SAMPLE POINTS (LFLT,LVERT) A SPREAD GIVING THE WIDTH
  OF THE GAUSSIAN PROFILE OR IF ZERO INDICATING A CONSTANT
  FLAT PROFILE (SFLT,SVERT) AND A MODE SELECTING REDUCED,
  EQUAL, OR INCREASED DIMENSION FOR IB WITH OPTIONAL RATIOING
  TO THE FILTER OUTPUT FOR CONSTANT INPUT (MFLT,MVERT). IF
  LVERT IS ZERO THE VERTICAL COMPONENT OF THE FILTER IS THE
  SAME AS THE HORIZONTAL. R(LR) IS THE ARRAY USED TO STORE
  THE LINES BEING PROCESSED.

SUBROUTINE PKFIND (R,LR,IA,IDEL,NMAX)
  FINDS THE HIGHEST PEAKS IN THE DATA ON IA, TO A MAXIMUM
  OF NMAX, USING A SPACING PARAMETER IDEL, AND LEAVING THE
  RESULTS IN /PKT/. R(LR) IS TEMPORARY LINE STORAGE.

SUBROUTINE PKINIT (NMX,IDEL,L,N,IA)
  INITIALISES /PKT/ BEFORE FINDING PEAKS IN ARRAY OF DATA.

SUBROUTINE PKPRO (RMAX,IMAX,IL)
  UPDATES /PKT/ USING THE PEAK GIVEN AS PARAMETERS.

SUBROUTINE PKIN (RMAX,IMAX,IL)
  INSERTS PEAK GIVEN AS PARAMETERS IN ORDERED TABLE /PKT/.

SUBROUTINE PKDEL (I)
  DELETES PEAK I FROM /PKT/.

SUBROUTINE PKSWP (IL)
  SWEEPS UP ALL FLAGGED PEAKS NOW OUT OF RANGE

SUBROUTINE PKLST (IOUT,IBOR)
  LISTS THE TABLE OF PEAKS IN /PKT/ TO UNIT IOUT. ZERO
  DEFAULTS TO 6 AND OUTPUTS THE INTEGER POSITIONS ONLY. IF
  IBOR IS NOT ZERO, RELATIVE POSITIONS WITHIN THE WINDOW ARE
  ALSO OUTPUT AND THOSE PEAKS WITHIN ARE SERIALLY NUMBERED.

SUBROUTINE PKCEN (R,LR,IA,IB,IBOR)
  LOCATES THE CENTRES OF THE PEAKS ALREADY FOUND BY
  PKFIND AND LISTS THEM TO IB. IF IA.LE.0 EXISTING
  DATA ONLY IS OUTPUT. IF IB IS 0 OUTPUT DEFAULTS TO
  6. IF IB IS -VE THERE IS NO OUTPUT AND RESULTS ARE
  LEFT IN COMMON /PKC/.

SUBROUTINE PEAKS (R,LR,IA,IDEL,NMAX,IBOR,IB,IC)
  FINDS PEAKS AND ESTIMATES CENTRES FROM IA. IF IA -VE IT
  OUTPUTS OLD DATA. IDEL, NMAX, CONTROL FINDING PEAKS,
  IBOR THE BORDER ON OUTPUT. IB SELECTS OUTPUT UNIT, 0
  DEFAULTING TO 6, AND -VE WITH NO OUTPUT OF CENTRE
  ESTIMATES. FINDING TABLE IS OUTPUT TO 6 FOR IB 6 OR -6.
  IF IC POSITIVE FULL PRINTER FORMATTED OUTPUT IS PRODUCED
  TO IC.
```

```
SUBROUTINE FLSUM (R,LR,S,LS,IA,IB,IC,SCALEA,SCALEB,ADDC,ITRIM)
  SUMS THE ARRAYS IN IA, IB IN THE AREA OF THEIR OVERLAP WITH
  WEIGHTS SCALEA, SCALEB AND A CONSTANT TERM ADDC, AND WRITES
  THE RESULT TO IC, AFTER TRIMMING OFF A BORDER ITRIM ON ALL
  SIDES. IA OR IB CAN BE ZERO.

SUBROUTINE FLLST (IA,IB,MPTS,MLINES,SCALE)
  LISTS THE FILE OF DATA IA TO IB AS A PRINTABLE
  ARRAY OF SMALL INTEGERS. MPTS,MLINES SET THE
  MAX NO OF PTS ON A LINE AND LINES OUTPUT, .LE.0
  SELECTING DEFAULTS 20,28 AND CAUSING -MPTS,
  -MLINES POINTS, LINES TO BE SKIPPED. IB 0
  DEFAULTS TO 6, IB -VE SELECTS -IB BUT WITH
  PRINTER FORMAT AND DEFAULT PARAMETERS 40,80.
  SCALE IS THE DATA SCALING FACTOR AND IF 0 IS
  SET AUTOMATICALLY FROM THE ENTIRE FILE OF DATA.

SUBROUTINE FLSTAT (R,LR,IA,BASE,RANGE,IUB,IC,BLOG)
  ANALYSES THE DISTRIBUTION OF DATA VALUES ON FILE
  IA AND OUTPUTS STATISTICS AND A DISTRIBUTION TO IUB AND
  IC IF NOT ZERO. DISTRIBUTION COVERS VALUES IN GIVEN RANGE
  ABOVE BASE. IF IUB IS 0, OUTPUT IS A SHORT FORM TO
  UNIT 6, IF -VE THERE IS NO OUTPUT BUT THE RESULTS ARE
  LEFT IN COMMON /HST/. IF IA IS 0 THE DATA IN /HST/ WILL
  BE OUTPUT. IF BASE AND RANGE ARE BOTH 0 ONLY THE
  STATISTICS ARE PRODUCED. IF RANGE ONLY IS 0 THE
  DISTRIBUTION HAS LOG INTERVALS. R(LR) IS THE LINE BUFFER.

SUBROUTINE FLDATA (IA,IU,IATYPE,LL,NL)
  RETURNS THE DIMENSIONS OF IMAGE IN FILE IA.

SUBROUTINE FLSET (IA,IB,INCL,INCN)
  SETS DIMENSIONS OF FILE IB FROM IA, REWINDS IA AND EMPTIES IB.

SUBROUTINE FLSIZE (IA,LL,NL)
  SETS DIMENSIONS FOR PRE-EXISTING FILE IN /FT/ AND
  REWINDS FILE UNLESS IA -VE.

SUBROUTINE GETLN (IA,DATA,LD,TEXT,LT,J,K)
  READS A LINE OF DATA FROM IA IF POSSIBLE, OTHERWISE RETURNS
  ZEROES. IA IS REWOUND AT EOF. TEXT(LT), J, K ARE DUMMIES.

SUBROUTINE PUTLN (IA,DATA,LD,TEXT,LT,J,K)
  WRITES A LINE OF DATA TO FILE IA. TEXT(LT), J, K ARE DUMMIES.
```

```
      SUBROUTINE STRGEN (IA,IB,ST,LST,NSTEL,D,LD,SPREAD,CUTOFF,MD)
C
C COPIES FROM FILE IA TO IB, ADDING STAR IMAGES AS SPECIFIED IN
C IN ARRAY ST(LST) WITH NSTEL ELEMENTS PER STAR, OF A PROFILE
C SELECTED BY MD AND SCALES TO A WIDTH SET BY SPREAD. A
C THRESHOLD LEVEL FOR OUTER PARTS IS SUPPLIED AS CUTOFF,
C ZERO DEFAULTING TO 0.0001. ARRAY D(LD) IS USED TO STORE
C THE LINE BEING PROCESSED. MD 4 OR 5 SELECTS EXTERNALLY
C GENERATED IMAGE COMPONENTS.
C
      REAL ST(LST),D(LD)
      INTEGER KL(1)
      DATA PI/3.1415926535898/
C
C***FIND SIZE OF SOURCE FILE ARRAY AND SET UP COPY FILE
      CALL FLDATA(IA,IU,IATYPE,LL,NL)
      CALL FLSET(IA,IB,0,0)
C***SET GRID PITCH (PROTECTED AGAINST LL.LE.1) AND LOCAL COPY
C***OF CUTOFF
      GRID=1.0/FLOAT(MAXO(1,(LL-1)))
      CUT=ABS(CUTOFF)
      IF(CUT.EQ.0.0) CUT=1.0E-4
C***FIND MAXIMUM BRIGHTNESS OF COMPONENTS
      BRMAX=0.0
      DO 90 IST=1,LST,NSTEL
      90.BRMAX=AMAX1(BRMAX,ABS(ST(IST+2)))
C***SEPARATE MODE PARAMETER FOR EXTERNAL IMAGE COMPONENTS
      MODE=MOD(MD,10)
      MCOMP=MD/10
C***SET CONSTANT FOR EFFECTIVE RADIUS SQUARED ACCORDING TO MODE
      SSQ=SPREAD**2
      RSQK=0.5*PI*SSQ
      IF(MODE.EQ.0) RSQK=2.0*SSQ
      IF(MODE.EQ.7) RSQK=SSQ
      IF(MODE.EQ.8) RSQK=((0.5*PI)**2)*SSQ
C***SET CONSTANT FOR PROFILE WIDTH ACCORDING TO MODE
      IF(SPREAD.NE.0.0) THEN
        PROFK=1.0/SSQ
        IF(MODE.EQ.0) PROFK=-0.5/SSQ
      ENDIF
C***INITIALISE ZONE FOR WHICH STARS CAN CONTRIBUTE TO A LINE
      RMAX=SQRT(RSQK)
      IF(MODE.EQ.0) RMAX=SQRT(RSQK*ALOG(AMAX1(1.0001,ABS(BRMAX/CUT))))
      IF(MODE.EQ.4) CALL ICOMPF(ST,LST,NSTEL,0,GRID,
     +  BRMAX,CUT,RMAX)
      IF(MODE.EQ.5) CALL ICOMPG(ST,LST,NSTEL,0,GRID,SPREAD,
     +  MCOMP,BRMAX,CUT,RMAX)
      IF(MODE.EQ.6) RMAX=GRID
      XL=0.0
      XMAX=RMAX
      IF(SPREAD.LE.0.0) XMAX=XMAX+0.5*GRID
      XMIN=-XMAX
C
C***LOOP OVER LINES
C
      DO 120 IL=1,NL
        CALL GETLN(IA,D,LL,KL,1,-1,K)
C
C  ***LOOP OVER STARS
C
        DO 110 IST=1,LST,NSTEL
C     ***ZONE CONTRIBUTING TO LINE HAS NOT REACHED STAR. NO ACTION
          IF(ST(IST).GT.XMAX) THEN
            IP=0
C     ***SINGLE POINT IMAGE ON NEAREST GRID POINT
          ELSEIF(SPREAD.EQ.0.0) THEN
            IP=INT(1.5+(ST(IST+1)/GRID))
            IF(IP.GT.0.AND.IP.LE.LL) D(IP)=D(IP)+ST(IST+2)
C       ***MOVE STAR OUT OF LIMITS
            ST(IST)=ST(IST)+1.0E6
C     ***ZONE HAS PASSED STAR, MOVE STAR OUT OF LIMITS
          ELSEIF(ST(IST).LE.XMIN) THEN
            ST(IST)=ST(IST)+1.0E6
C     ***STAR IN ZONE, NORMAL IMAGE
          ELSE
C       ***ADJUST STAR COORDINATES TO NEAREST GRID POINT IF REQ.
            XS=ST(IST)
            YS=ST(IST+1)
            IF(SPREAD.LT.0.0) THEN
              XS=GRID*(FLOAT(INT((XS/GRID)+1000000.5))-1000000.0)
              YS=GRID*(FLOAT(INT((YS/GRID)+1000000.5))-1000000.0)
            ENDIF
C       ***FIND MAXIMUM EFFECTIVE RADIUS SQUARED FOR STAR
            IF(MODE.EQ.4) THEN
              CALL ICOMPF(ST,LST,NSTEL,-IST,GRID,
     +          ST(IST+2),CUT,RMAX)
              RSQMAX=RMAX**2
            ELSEIF(MODE.EQ.5) THEN
              CALL ICOMPG(ST,LST,NSTEL,-IST,GRID,SPREAD,
     +          MCOMP,ST(IST+2),CUT,RMAX)
              RSQMAX=RMAX**2
            ELSEIF(ABS(ST(IST+2)).GT.CUT) THEN
              RSQMAX=RSQK
              IF(MODE.EQ.0) RSQMAX=RSQK*ALOG(ABS(ST(IST+2)/CUT))
            ELSE
              RSQMAX=-1.0
            ENDIF
C       ***MINIMUM DISTANCE SQUARED FROM LINE TO STAR
            DX=XL-XS
            DXSQ=DX**2
C       ***SPECIAL CASE OF MIN SIZE EXACT POSITION IMAGE
            IF(MODE.EQ.6) THEN
              WLN=AMAX1(0.0,(1.0-ABS(DX/GRID)))
              IPL=INT(YS/GRID)+1
              IPR=IPL+1
              WPL=AMAX1(0.0,(1.0-ABS((YS/GRID)-FLOAT(IPL-1))))
              WPR=AMAX1(0.0,(1.0-ABS((YS/GRID)-FLOAT(IPR-1))))
              IF(IPL.GE.1.AND.IPL.LE.LL) D(IPL)=
     +          D(IPL)+WLN*WPL*ST(IST+2)
              IF(IPR.GE.1.AND.IPR.LE.LL) D(IPR)=
     +          D(IPR)+WLN*WPR*ST(IST+2)
              RSQMAX=-1.0
            ENDIF
C       ***SKIP TO NEXT STAR IF THIS ONE TOO FAR FROM LINE
            IF(DXSQ.LE.RSQMAX) THEN
C
C         ***LOOP OVER LINE FOR THIS LINE/STAR PAIR
C
C         ***DEFINE POINTS OF LINE WITHIN RMAX OF YS
              IPL=INT(((YS-RMAX)/GRID)+0.9999)+1
              IPR=INT((YS+RMAX)/GRID)+1
              IPL=MAXO(1,MINO(LL,IPL))
              IPR=MAXO(1,MINO(LL,IPR))
              IPR=MAXO(IPL,IPR)
C
              DO 100 IP=IPL,IPR
                DY=FLOAT(IP-1)*GRID-YS
                RSQ=DXSQ+DY**2
C           ***SKIP PROCESSING IF POINT NOT WITHIN EFFECTIVE
C           ***RADIUS OF A STAR
                IF(RSQ.LE.RSQMAX) THEN
                  IF(MODE.EQ.0) THEN
C                   ***GAUSSIAN
                    D(IP)=D(IP)+ST(IST+2)*EXP(PROFK*RSQ)
                  ELSEIF(MODE.EQ.4) THEN
C                   ***EXTERNAL COMPONENT F
                    CALL ICOMPF(ST,LST,NSTEL,IST,GRID,
     +                DX,DY,A)
                    D(IP)=D(IP)+ST(IST+2)*A
                  ELSEIF(MODE.EQ.5) THEN
C                   ***EXTERNAL COMPONENT G
                    CALL ICOMPG(ST,LST,NSTEL,IST,GRID,SPREAD,
     +                MCOMP,DX,DY,A)
                    D(IP)=D(IP)+ST(IST+2)*A
                  ELSEIF(MODE.EQ.7) THEN
C                   ***UNIFORM DISC
                    D(IP)=D(IP)+ST(IST+2)
                  ELSEIF(MODE.EQ.8) THEN
C                   ***COS SQUARED
                    D(IP)=D(IP)+ST(IST+2)*(COS(SQRT(PROFK*RSQ))**2)
                  ELSE
C                   ***COS SQUARED OF R SQUARED (DEFAULT)
                    D(IP)=D(IP)+ST(IST+2)*(COS(PROFK*RSQ)**2)
                  ENDIF
                ENDIF
              100 CONTINUE
C
            ENDIF
          ENDIF
        110 CONTINUE
C
        CALL PUTLN(IB,D,LL,KL,1,-1,K)
C  ***INCREMENT X VALUES FOR NEXT LINE
        XL=XL+GRID
        XMAX=XMAX+GRID
        XMIN=XMIN+GRID
      120 CONTINUE
C
C***RESTORE TABLE
C
      DO 130 IST=1,LST,NSTEL
        IF(ST(IST).GT.0.5E6) ST(IST)=ST(IST)-1.0E6
        IF(ST(IST+1).GT.0.5E6) ST(IST+1)=ST(IST+1)-1.0E6
      130 CONTINUE
C
      RETURN
      END
C*** OF STRGEN
```

```
      SUBROUTINE ICOMPF (ST,LST,NSTEL,IST,GRID,X,Y,OUT)
C
C RETURNS AN INTENSITY POINT ON AN ELLIPTICAL GALAXY IMAGE,
C SELECTED FROM TABLE ST(LST) BY IST, AT POSITION X,Y.
C FOR IST NEGATIVE IT RETURNS THE MAXIMUM RADIUS OF THE
C GALAXY -IST. FOR IST ZERO IT PERFORMS INITIALISING
C CONVERSIONS ON TABLE AND RETURNS THE OVERALL MAXIMUM
C RADIUS. GIVEN FRACTION OF FLUX THAT CAN BE CUT OFF AS Y.
C
      REAL ST(LST)
C
      IF(IST.EQ.0) THEN
C  ***INITIALISE. RETURN OVERALL MAX RADIUS
        CALL INITCF(ST,LST,NSTEL,GRID,Y,OUT)
      ELSEIF(IST.LT.0) THEN
C  ***RETURN MAX RADIUS
        OUT=ST(-IST+6)
      ELSEIF(ST(IST+4).LE.1.0E-4*GRID) THEN
C  ***STAR
        DX=ABS(X)
        DY=ABS(Y)
        IF(DX.LT.GRID.AND.DY.LT.GRID) THEN
          OUT=(1.0-DX/GRID)*(1.0-DY/GRID)
        ELSE
          OUT=0.0
        ENDIF
      ELSE
C  ***GALAXY
        DX=(ST(IST+8)*X+ST(IST+9)*Y)*ST(IST+10)
        DY=(ST(IST+8)*Y-ST(IST+9)*X)*ST(IST+11)
        RSQ=AMAX1((DX**2+DY**2),0.1*(GRID**2))
        RQ=(RSQ/(ST(IST+4)**2))**0.125-1.0
        OUT=ST(IST+7)*EXP(-7.67*RQ)
        IF(AMAX1(ABS(X),ABS(Y)).LT.0.2*GRID) THEN
C    ***CENTRE POINT
          REQ=GRID/1.77
          Z=(REQ/ST(IST+4))**0.25
          F=(1.035/(1.0+EXP(5.0*(1.0-Z))))-0.035
          OUT=AMAX1(OUT,F)
        ENDIF
      ENDIF
C
      RETURN
      END
C*** OF ICOMPF
C
C
C
      SUBROUTINE INITCF (ST,LST,NSTEL,GRID,CUT,OUT)
C
C INITIALISES TABLE ST(LST) OF ELLIPTICAL GALAXIES.
C
      REAL ST(LST)
C
C***ZERO OVERALL MAXIMUM RADIUS
      RMAX=0.0
C***CALCULATE RADIUS CUTOFF FACTOR
      RCUT=2.3/(0.015+SQRT(AMAX1(1.0E-6,AMIN1(0.2,CUT))))
C
C***LOOP THROUGH TABLE
C
      DO 100 I=1,LST,NSTEL
C  ***SET COS AND SIN OF POSITION ANGLE
        A=ST(I+3)*3.14159/180.0
        ST(I+8)=COS(A)
        ST(I+9)=SIN(A)
C  ***SET COMPRESSION FACTORS IN X AND Y
        ST(I+10)=SQRT(1.0-AMIN1(0.99,ST(I+5)))
        ST(I+11)=1.0/ST(I+10)
C  ***SET MAXIMUM OBJECT RADIUS AND AMPLITUDE
        IF(ST(I+4).GT.1.0E-4*GRID) THEN
C    ***GALAXY
          ST(I+6)=ST(I+4)*ST(I+11)*RCUT
          ST(I+7)=((GRID/ST(I+4))**2)/(7.22*3.14159)
        ELSE
C    ***STAR
          ST(I+6)=2.0*GRID
          ST(I+7)=0.0
        ENDIF
C  ***UPDATE OVERALL MAXIMUM RADIUS
        RMAX=AMAX1(RMAX,ST(I+6))
  100 CONTINUE
C
      OUT=RMAX
      RETURN
      END
C*** OF INITCF
      SUBROUTINE GRDCLR (S,LS,NSTEL,NS,NR,NC,NROW,NCOL,SEED)
C
C CLEARS TABLE S(LS), SETS NUMBER OF ROWS AND COLUMNS TO
C NROW,NCOL AND NS TO TOTAL NUMBER OF ELEMENTS. RANDOM
C NUMBER GENERATOR IS INITIALISED UNLESS SEED IS NEGATIVE.
C
      REAL S(LS)
      COMMON/ERR/JERR,IERR
C
      IF(NCOL.LE.0.OR.NROW.LE.0.OR.(NSTEL*NROW*NCOL).GT.LS) THEN
        IF(IERR.GT.0) WRITE(IERR,*) ' PARAMETER ERROR'
        RETURN
      ENDIF
C
      NS=NCOL*NROW*NSTEL
      NC=NCOL
      NR=NROW
C
      DO 100 I=1,LS
  100 S(I)=0.0
C
      IF(SEED.GE.0) CALL RDG(SEED,1)
C
      RETURN
      END
C*** OF GRDCLR
C
C
C
      SUBROUTINE GRDDEF (S,LS,NSTEL,NR,NC,XL,XH,YL,YH,AMP)
C
C ENTERS A RECTANGULAR GRID OF COORDINATES IN TABLE S(LS),
C X FROM XL TO XH, Y FROM YL TO YH, AND ADDS RANDOM VALUES
C IN THE RANGE -AMP TO +AMP TO EACH COORDINATE.
C
      REAL S(LS)
C
      CALL GRDSET(S,LS,NSTEL,NR,NC,1,12,
     +  XL,(XH-XL)/FLOAT(MAXO(1,NR-1)),0.0)
      CALL GRDSET(S,LS,NSTEL,NR,NC,2,11,
     +  YL,(YH-YL)/FLOAT(MAXO(1,NC-1)),0.0)
      IF(AMP.NE.0.0) THEN
        CALL GRDSET(S,LS,NSTEL,NR,NC,1,3,-AMP,AMP,0.0)
        CALL GRDSET(S,LS,NSTEL,NR,NC,2,3,-AMP,AMP,0.0)
      ENDIF
C
      RETURN
      END
C*** OF GRDDEF
C
C
C
      SUBROUTINE GRDDBS (S,LS,NSTEL,NR,NC,JCON)
C
C CONVERTS TABLE S(LS) FROM DOUBLE STARS TO INDIVIDUAL STARS,
C EACH PAIR OF THE GIVEN BRIGHTNESS AND THIS MULTIPLIED BY THE
C GIVEN RATIO. IF FLAG JCON IS +VE, THE TOTAL BRIGHTNESS IS
C MADE EQUAL TO THE GIVEN VALUE.
C
      REAL S(LS)
C
      DO 100 I=1,NR*NC
        K=(I-1)*NSTEL
        A=(3.1415926535/180.0)*S(K+4)
        SEP=S(K+8)
        BR=S(K+3)
        RAT=S(K+7)
        IF(RAT.EQ.-1.0) RAT=RAT+1.0E-6
        RATA=1.0/(1.0+RAT)
        RATB=RAT*RATA
C
        IF(JCON.LE.0) THEN
          BRA=BR
          BRB=BR*RAT
        ELSE
C    ***FIXED TOTAL BRIGHTNESS
          BRA=BR*RATA
          BRB=BR*RATB
        ENDIF
C
        S(K+3)=BRA
        S(K+7)=BRB
C
        DX=SEP*COS(A)
        DY=SEP*SIN(A)
        S(K+5)=S(K+1)-RATA*DX
        S(K+6)=S(K+2)-RATA*DY
        S(K+1)=S(K+5)+DX
        S(K+2)=S(K+6)+DY
  100 CONTINUE
C
      NSTEL=4
C
      RETURN
      END
C*** OF GRDDBS
```

```
      SUBROUTINE GRDSET (S,LS,NSTEL,NR,NC,IP,MD,X,Y,Z)
C
C SETS VALUES OF ELEMENT IP (OUT OF NSTEL ELEMENTS) IN TABLE
C S(LS) ACCORDING TO MODE MD AND THREE INPUT PARAMETERS.
C
      REAL S(LS)
C
      MS=MD/10
      MF=MOD(MD,10)
      IPT=MAXO(1,MINO(NSTEL,IP))
C
      IF(MD.EQ.3.AND.Z.GE.0.0) THEN
C  ***SET RANDOM NUMBER SEED AND RETURN IF ZERO AMPLITUDE
        IF(Z.GT.0.0.OR.X.EQ.Y) CALL RDG(Z,1)
        IF(X.EQ.Y) RETURN
      ENDIF
C
C***SET LOOPING PARAMETERS
C
      IF(MF.EQ.1) THEN
C  ***STEP ALONG ROW
        NSTEP=NC
        INCS=1
        NREP=NR
        INCR=NC
      ELSEIF(MF.EQ.2) THEN
C  ***STEP DOWN COLUMN
        NSTEP=NR
        INCS=NC
        NREP=NC
        INCR=1
      ELSE
C  ***STEP THROUGH ALL POINTS
        NSTEP=NC*NR
        INCS=1
        NREP=1
        INCR=0
      ENDIF
C***SET NUMBER OF ELEMENTS FOR EACH INCREMENT
      NGRP=MAXO(1,INT(Z))
C***SPECIAL CASE OF SINGLE ROW OR COLUMN
      LREP=1
      IF(MD.EQ.1.OR.MD.EQ.2) THEN
        LREP=MAXO(1,MINO(NREP,INT(Y)))
        NREP=LREP
      ENDIF
C
C
C
      DO 110 I=LREP,NREP
        A=X
        K=(I-1)*INCR-INCS
        DO 100 J=1,NSTEP
          K=K+INCS
          L=IPT+NSTEL*K
C
          IF(MS.EQ.0) THEN
            IF(MF.EQ.0) THEN
C             ***ADD TO OR ZERO ALL POINTS
              S(L)=S(L)+X
              IF(X.EQ.0.0) S(L)=0.0
            ELSEIF(MF.LE.2) THEN
C             ***SET ROW OR COLUMN
              S(L)=X
            ELSEIF(MF.EQ.3) THEN
C             ***ADD RANDOM NUMBER BETWEEN X AND Y
              CALL RDG(R,-1)
              S(L)=S(L)+R*(Y-X)+X
            ELSEIF(MF.EQ.4) THEN
C             ***ANTILOG
              S(L)=10.0**S(L)
              IF(X.NE.0.0) S(L)=S(L)*X
            ELSEIF(MF.EQ.5) THEN
C             ***SCALE
              S(L)=X*S(L)+Y
            ENDIF
          ELSE
            S(L)=S(L)+A
            MJ=MOD(J,NGRP)
            IF(MJ.EQ.0.AND.MS.EQ.2) THEN
C             ***GEOMETRIC SERIES
              A=A*Y
            ELSEIF(MJ.EQ.0) THEN
C             ***ARITHMETIC SERIES
              A=A+Y
            ENDIF
          ENDIF
C
  100   CONTINUE
  110 CONTINUE
C
      RETURN
      END
C*** OF GRDSET
      SUBROUTINE GRDRND (S,LS,NSTEL,NR,NC,IP)
C
C RANDOMISES THE ORDER OF ELEMENTS IN POSITION IP IN TABLE S(LS).
C
      REAL S(LS)
C
      IPT=MAXO(1,MINO(NSTEL,IP))
      NEL=NR*NC
C
      DO 100 I=1,5*NEL
        CALL RDG(RA,-1)
        CALL RDG(RB,-1)
        IA=MINO(NEL-1,INT(RA*FLOAT(NEL)))
        IB=MINO(NEL-1,INT(RB*FLOAT(NEL)))
        KA=IPT+NSTEL*IA
        KB=IPT+NSTEL*IB
        T=S(KA)
        S(KA)=S(KB)
        S(KB)=T
  100 CONTINUE
C
      RETURN
      END
C*** OF GRDRND
C
C
C
      SUBROUTINE STRCLR (NS)
C
C CLEARS THE TABLE OF STARS BY RESETTING THE COUNT OF
C ELEMENTS TO ZERO.
C
      NS=0
      RETURN
      END
C*** OF STRCLR
C
C
C
      SUBROUTINE STRSET (ST,LST,NSTEL,NS,R,LR)
C
C ENTERS NSTEL VALUES TAKEN FROM R(LR) INTO THE NEXT
C PLACES IN ST(LST) AND ADVANCES NS TO LAST ELEMENT.
C
      REAL ST(LST),R(LR)
C
      IF(NSTEL.LE.0.OR.NSTEL.GE.LR.OR.NS.LT.0.OR.
     +  NS.GT.(LST-NSTEL)) RETURN
C
      DO 110 I=1,NSTEL
        ST(NS+I)=R(1+I)
  110 CONTINUE
      NS=NS+NSTEL
C
      RETURN
      END
C*** OF STRSET
C
C
C
      SUBROUTINE STRMOD (ST,LST,NSTEL,R,LR)
C
C ALTERS AN ENTRY IN ST(LST) SELECTED BY THE FIRST
C DATA VALUE TO CONTAIN THE FOLLOWING DATA VALUES.
C
      REAL R(LR),ST(LST)
C
      N=IFIX(R(2))
      IF((N*NSTEL).LE.LST.AND.N.GE.1.AND.NSTEL.LE.(LR-2)) THEN
        DO 100 I=1,NSTEL
  100   ST((N-1)*NSTEL+I)=R(I+2)
      ENDIF
C
      RETURN
      END
C*** OF STRMOD
C
C
C
      SUBROUTINE STRLST (IU,ST,LST,NSTEL,NS)
C
C LISTS THE TABLE ST(LST) UP TO ELEMENT NS IN LINES
C OF NSTEL VALUES, LINES NUMBERED.
C
      REAL ST(LST)
C
      IF(IU.LT.0.OR.NSTEL.LE.0.OR.NS.LE.0.OR.NS.GT.LST) RETURN
C
      IUA=IU
      IF(IU.EQ.0) IUA=6
C
      DO 110 I=1,NS,NSTEL
        WRITE(IUA,1100)
     +    ((I/NSTEL)+1),(ST(IS),IS=I,(I+NSTEL-1))
 1100   FORMAT (1X,I3,1H/,6F10.4,(/5X,6F10.4))
  110 CONTINUE
C
      RETURN
      END
C*** OF STRLST
```

```
      SUBROUTINE STRFLD (NS,GAP,BGAP,SEED,SHAPE)
C
C CLEARS STAR TABLE, SETS GAP REQUIRED BETWEEN STARS
C IN /STRCOM/ AND BORDER GAP, AND SHAPE OF FIELD (0 DEFAULTS
C TO 1), AND INITIALISES RANDOM NUMBER GENERATOR.
C
      COMMON/STRCOM/STRGAP,XLIM,BORGAP,RSTR(5),KSTR(8)
C
      NS=0
      STRGAP=GAP
      BORGAP=BGAP
      XLIM=SHAPE
      IF(XLIM.LE.0.0) XLIM=1.0
      CALL RDG(SEED,1)
C
      RETURN
      END
C*** OF STRFLD
C
C
C
      SUBROUTINE STRTST (ST,LST,NSTEL,NS,NTEST,JGAP)
C
C TESTS LAST NTEST ENTRIES IN ST(LST) OUT OF THE ENTRIES
C EACH OF NSTEL ELEMENTS (TOTAL NS ELEMENTS) FOR DISTANCE
C FROM ALL PRECEDING ENTRIES AND FROM EDGE. JGAP IS SET TO 1
C IF SPACING IS AT LEAST STRGAP,BORGAP RESPECTIVELY, OTHERWISE 0.
C
      REAL ST(LST)
      COMMON/STRCOM/STRGAP,XLIM,BORGAP,RSTR(5),KSTR(8)
C
      NSTR=NS/NSTEL
      XMAX=XLIM-BORGAP
      XMIN=BORGAP
      YMAX=1.0-BORGAP
      YMIN=BORGAP
      GAPSQ=STRGAP**2
C
      JGAP=0
      DO 110 IT=(MAXO(0,NSTR-NTEST)+1),NSTR
        XT=ST((IT-1)*NSTEL+1)
        YT=ST((IT-1)*NSTEL+2)
        IF(XT.LT.XMIN.OR.XT.GT.XMAX.OR.
     +     YT.LT.YMIN.OR.YT.GT.YMAX) RETURN
        IF(NSTR.GT.NTEST) THEN
          DO 100 IS=1,(NSTR-NTEST)
            XS=ST((IS-1)*NSTEL+1)
            YS=ST((IS-1)*NSTEL+2)
            IF(((XS-XT)**2+(YS-YT)**2).LT.GAPSQ) RETURN
  100     CONTINUE
        ENDIF
  110 CONTINUE
C
      JGAP=1
      RETURN
      END
C*** OF STRTST
C
C
C
      SUBROUTINE STRINS (ST,LST,NSTEL,NS,BRIGHT,NUM)
C
C ADDS NUM (0 DEFAULTS TO 1) STARS OF BRIGHTNESS BRIGHT TO
C STAR TABLE ST(LST) (ENTRIES OF NSTEL ELEMENTS, TOTAL NS)
C IN RANDOM POSITIONS BUT REQUIRING A GAP STRGAP BETWEEN
C STARS AND BORGAP AT EDGE. AFTER 100 FAILURES THE ATTEMPT
C WILL BE ABANDONED.
C
      REAL ST(LST)
      COMMON/STRCOM/STRGAP,XLIM,BORGAP,RSTR(5),KSTR(8)
      COMMON/ERR/JERR,IERR
C
      N=MAXO(1,NUM)
      IFAIL=0
C
      DO 120 I=1,N
        IF((NS+NSTEL).GT.LST) THEN
          IF(IERR.GT.0) WRITE(IERR,*) ' STAR TABLE FULL'
          RETURN
        ENDIF
C
        NS=NS+NSTEL
        100 CONTINUE
          CALL RDG(X,-1)
          CALL RDG(Y,-1)
          KS=NS-NSTEL
          ST(KS+1)=X*XLIM
          ST(KS+2)=Y
          ST(KS+3)=BRIGHT
C         ***ZERO ANY EXTRA ELEMENTS IN TABLE
          IF(NSTEL.GT.3) THEN
            DO 110 IS=4,NSTEL
  110       ST(KS+IS)=0.0
          ENDIF
C         ***TEST FOR SPACING
          CALL STRTST(ST,LST,NSTEL,NS,1,JGAP)
          IF(JGAP.LE.0) IFAIL=IFAIL+1
          IF(IFAIL.GT.100) THEN
            NS=NS-NSTEL
            IF(IERR.GT.0) WRITE(IERR,1115) NS/NSTEL
 1115       FORMAT(' FIELD FULL,',I4,' ENTRIES')
            RETURN
          ENDIF
        IF(JGAP.LE.0) GO TO 100
  120 CONTINUE
C
      RETURN
      END
C*** OF STRINS
C
C
C
      SUBROUTINE STRDBL (ST,LST,NSTEL,NS,FAINT,RATIO,SEP,NUM)
C
C ADDS NUM (0 DEFAULTS TO 1) DOUBLE STARS TO TABLE ST(LST)
C (ENTRIES OF NSTEL ELEMENTS, TOTAL NS). FAINT GIVES THE
C FAINTER COMPONENT, THE OTHER BEING GREATER BY A FACTOR
C RATIO. SEP IS GIVEN BUT PA IS RANDOM, AND IS WRITTEN TO
C THE 4TH ELEMENT OF THE FIRST ENTRY IF THERE IS ROOM (AS
C POSITION ANGLE OF WEAKER COMPONENT ANTICLOCKWISE FROM
C X AXIS). AFTER 100 FAILURES THE ATTEMPT WILL BE ABANDONED.
C
      REAL ST(LST)
      COMMON/STRCOM/STRGAP,XLIM,BORGAP,RSTR(5),KSTR(8)
      COMMON/ERR/JERR,IERR
C
      N=MAXO(1,NUM)
      RAT=AMAX1(1.0,RATIO)
      BRIGHT=RAT*FAINT
      IFAIL=0
C
      DO 130 I=1,N
        IF((NS+2*NSTEL).GT.LST) THEN
          IF(IERR.GT.0) WRITE(IERR,*) ' STAR TABLE FULL'
          RETURN
        ENDIF
C
        NS=NS+2*NSTEL
        CALL RDG(PA,-1)
        A=6.283185*PA
        DEG=360.0*PA
        DX=SEP*COS(A)
        DY=SEP*SIN(A)
        DXB=DX/(1.0+RAT)
        DXF=DXB*RAT
        DYB=DY/(1.0+RAT)
        DYF=DYB*RAT
C
        100 CONTINUE
          CALL RDG(X,-1)
          CALL RDG(Y,-1)
          KS=NS-2*NSTEL
          ST(KS+1)=X*XLIM+DXF
          ST(KS+2)=Y+DYF
          ST(KS+3)=FAINT
          IF(NSTEL.GT.3) THEN
            DO 110 IS=4,NSTEL
  110       ST(KS+IS)=0.0
            ST(KS+4)=DEG
          ENDIF
          KS=KS+NSTEL
          ST(KS+1)=X*XLIM-DXB
          ST(KS+2)=Y-DYB
          ST(KS+3)=BRIGHT
          IF(NSTEL.GT.3) THEN
            DO 120 IS=4,NSTEL
  120       ST(KS+IS)=0.0
          ENDIF
C         ***TEST FOR SPACING
          CALL STRTST(ST,LST,NSTEL,NS,2,JGAP)
          IF(JGAP.LE.0) IFAIL=IFAIL+1
          IF(IFAIL.GT.100) THEN
            NS=NS-2*NSTEL
            IF(IERR.GT.0) WRITE(IERR,1125) NS/NSTEL
 1125       FORMAT(' FIELD FULL,',I4,' ENTRIES')
            RETURN
          ENDIF
        IF(JGAP.LE.0) GO TO 100
  130 CONTINUE
C
      RETURN
      END
C*** OF STRDBL
```

```
 SUBROUTINE NOISE (IA,IB,MODE,D,LD,A,B,SEED)
C
C ADDS RANDOM NOISE TO FILE IA, COPYING TO IB. TYPE OF NOISE
C IS SPECIFIED BY MODE, WITH PARAMETERS A AND B, DERIVED FROM
C INITIAL VALUE SEED. ARRAY D(LD) IS TEMPORARY STORAGE FOR
C LINE.
C
 REAL D(LD)
 INTEGER KL(1)
C
C***SET UP COPY FILE AND OBTAIN SIZE
 CALL FLSET(IA,IB,0,0)
 CALL FLDATA(IA,IU,IATYPE,L,N)
C
C***LOOP OVER LINES
C
 DO 110 ILN=1,N
   CALL GETLN(IA,D,L,KL,1,-1,K)
C  ***INITIALISE RDG FOR THIS LINE
   CALL RDG(SEED,ILN)
C  ***LOOP OVER POINTS
   DO 100 IP=1,L
C    ***GENERATE AND ADD NOISE ACCORDING TO MODE
     CALL RDG(DN,0)
     IF(MODE.LE.1) THEN
       T=D(IP)+(DN*A)
     ELSE
       T=D(IP)+(DN*A*SQRT(AMAX1(0.0,(D(IP)+B))))
     ENDIF
     D(IP)=T
   100 CONTINUE
C  ***REPLACE LINE OF DATA
   CALL PUTLN(IB,D,L,KL,1,-1,K)
 110 CONTINUE
C
 RETURN
 END
C*** OF NOISE
```

```
 SUBROUTINE RDG (Z,N)
C
C RANDOM DEVIATE GENERATOR. RETURNS VALUE Z, NORMAL DEVIATE
C FOR N=0, UNIFORM FOR N=-1, RANDOM SIGN FOR N=-2. N GREATER
C THAN 0 CAUSES INITIALISATION TO A POINT IN SEQUENCE
C SELECTED BY N AND BY Z. Z IS IN RANGE 0.0 ... 1.0.
C
 INTEGER IT(24)
 DATA (IT(I),I=1,24)/-16,-6,-8,-11,-1,-15,-20,-4,-10,
+  -14,-18,-23,-13,-17,-2,-7,-22,-12,-3,-5,-9,0,-19,-21/
 DATA IA,IB/9.3217320416231678/
 DATA IC,ID/7.526003142654331B/
 DATA IK,IL/777777777777777B,1000000000000000B/
 DATA TPI,DEL,SEED/6.283185307,1.0E-12,0.0/
 DATA IS,IX,NOLD,LN/0,0,0,999999/
C
 NXT(IX)=IK.AND.(IX+SHIFT(IX,IA)+IB)
 SCALE(IX)=FLOAT(SHIFT(IX,-3))/FLOAT(IL)
C
C***RESET STATUS IF IN DIFFERENT MODE TO PREVIOUS CALL
 IF(N.NE.NOLD) IS=0
 NOLD=N
C
 IF(N.GT.0) THEN
C  ***SET UP
   IF(Z.NE.SEED.OR.N.LT.LN) THEN
C    ***FROM BEGINNING IF NEW SEED OR EARLIER LINE
     SEED=Z
     LN=-10
     K=IFIX(1.3172E6*Z)*2649
     ILN=0B
     DO 100 I=1,24
       ILN=SHIFT(ILN,1)
       IF((SHIFT(K,IT(I)).AND.1).NE.0) ILN=ILN+1
     100 CONTINUE
   ENDIF
   IF(N.GT.LN) THEN
C    ***ADVANCE TO GIVEN LINE
     DO 110 I=(LN+1),N
     110 ILN=IK.AND.(ILN+SHIFT(ILN,IC)+ID)
     LN=N
   ENDIF
   IX=ILN
   IS=0
 ELSEIF(N.EQ.0) THEN
C  ***NORMAL DEVIATES
   IF(IS.EQ.0) THEN
C    ***FIRST VALUE OF A PAIR
     IX=NXT(IX)
     UA=SCALE(IX)
     IX=NXT(IX)
     UB=SCALE(IX)
     A=SQRT(4.*DEL-2.*ALOG(UA+DEL))
     Z=A*SIN(TPI*UB)
     IS=1
   ELSE
     Z=A*COS(TPI*UB)
     IS=0
   ENDIF
 ELSEIF(N.EQ.-1) THEN
C  ***UNIFORM DEVIATES
   IX=NXT(IX)
   Z=SCALE(IX)
   IS=0
 ELSE
C  ***RANDOM SIGN, +/-1.0
   IX=NXT(IX)
   Z=-1.0
   IF(IX.GT.(IL*4)) Z=1.0
   IS=0
 ENDIF
C
 RETURN
 END
C*** OF RDG
```

```
      SUBROUTINE FLT (R,LR,IA,IFLT,IB)
C
C FILTERS ARRAY OF DATA FROM IA ONTO IB BY CONVOLVING
C WITH THE ARRAY FROM FILE IFLT. R(LR) IS THE ARRAY USED
C TO STORE THE FILTER AND THE LINES BEING PROCESSED.
C
      REAL R(LR)
      INTEGER KL(1)
      COMMON/ERR/JERR,IERR
C
      CALL FLDATA(IA,IU,IATYPE,LL,NL)
      CALL FLSET(IA,0,0,0)
      CALL FLDATA(IFLT,IUF,IFTYPE,LFLTH,LFLTV)
      CALL FLSET(IFLT,0,0,0)
C
C***SET OUTPUT AND BUFFER LINE LENGTHS
      LLB=LL-(LFLTH-1)
      LLR=LL
C***SET OFFSETS IN BUFFER ARRAY
      KBUFF=LLR
      KFLTV=KBUFF+LFLTV*LLR
      KFLTH=KFLTV+LFLTV+1
C***SET OUTPUT LINE LIMITS AND THEIR NUMBER
      NLFIRST=1+(LFLTV-1)
      NLLAST=NL
      NLB=NL-(LFLTV-1)
C
C***RETURN IF OUT OF SPACE OR DIMENSIONS NOT POSITIVE
C
      IF((KFLTH+LFLTH*LFLTV).GT.LR.OR.LLB.LE.0.OR.NLB.LE.0.
     +  OR.LFLTV.LE.0.OR.LFLTH.LE.0) THEN
        IF(IERR.GT.0) WRITE(IERR,*) ' OUT OF SPACE'
        RETURN
      ENDIF
C
      CALL FLSET(-IATYPE,IB,LLB,NLB)
C
C***READ IN FILTER
      DO 110 IFV=1,LFLTV
        CALL GETLN(IFLT,R,LFLTH,KL,1,-1,K)
C  ***CALCULATE OFFSET FOR LINE OF FILTER
        KFT=KFLTH+(IFV-1)*LFLTH
        R(KFLTV+IFV)=FLOAT(KFT)
C  ***COPY LINE OF FILTER INTO PLACE
        DO 100 IFH=1,LFLTH
        100 R(KFT+IFH)=R(IFH)
      110 CONTINUE
C
C***ZERO BUFFER AND SET POINTERS
C
      DO 130 I=1,(LLR*LFLTV)
      130 R(KBUFF+I)=0.0
      LINENO=1
      KNXTL=KBUFF
C
C***LOOP OVER LINES
C
      DO 260 IL=1,NLLAST
C
C  ***ZERO NEXT LINE
C
        DO 140 I=1,LLR
        140 R(KNXTL+I)=0.0
C
C  ***ENTER NEXT LINE IF DATA, AND COPY INTO PLACE
C
        IF(IL.LE.NL) THEN
          CALL GETLN(IA,R,LL,KL,1,-1,K)
          DO 160 I=1,LL
          160 R(KNXTL+I)=R(I)
        ENDIF
C
C  ***SCROLL DOWN VERTICAL FILTER INTO CORRECT POSITION
C
        DO 200 I=1,LFLTV
        200 R(KFLTV+2+LFLTV-I)=R(KFLTV+1+LFLTV-I)
        R(KFLTV+1)=R(KFLTV+LFLTV+1)
C
C  ***OUTPUT LINE IF REQUIRED
C
        IF(IL.GE.NLFIRST.AND.IL.LE.NLLAST) THEN
C         ***ZERO OUTPUT LINE
          DO 210 I=1,LLB
          210 R(I)=0.0
C         ***FILTER
          DO 240 ILH=1,LLB
            S=0.0
            DO 230 IFV=1,LFLTV
              KLN=KBUFF+(IFV-1)*LLR+(ILH-1)
              KF=INT(R(KFLTV+IFV))
              DO 220 IFH=1,LFLTH
              220 S=S+R(KF+IFH)*R(KLN+IFH)
            230 CONTINUE
            R(ILH)=S
          240 CONTINUE
C         ***WRITE OUT LINE
          CALL PUTLN(IB,R,LLB,KL,1,-1,K)
        ENDIF
C
C  ***ADVANCE POINTER TO NEXT LINE
C
        LINENO=LINENO+1
        IF(LINENO.GT.LFLTV) LINENO=1
        KNXTL=KBUFF+(LINENO-1)*LLR
C
      260 CONTINUE
C
      RETURN
      END
C*** OF FLT
C
C
C
      SUBROUTINE FLTXY (R,LR,IA,IB,LFLT,SFLT,MFLT,LVERT,SVERT,MVERT)
C
C FILTERS ARRAY OF DATA FROM IA ONTO IB. APPLYING SEPARATE
C X AND Y COMPONENTS. EACH IS SPECIFIED BY A LENGTH IN TERMS
C OF SAMPLE POINTS (LFLT,LVERT) A SPREAD GIVING THE WIDTH
C OF THE GAUSSIAN PROFILE OR IF ZERO INDICATING A CONSTANT
C FLAT PROFILE (SFLT,SVERT) AND A MODE SELECTING REDUCED,
C EQUAL, OR INCREASED DIMENSION FOR IB WITH OPTIONAL RATIOING
C TO THE FILTER OUTPUT FOR CONSTANT INPUT (MFLT,MVERT). IF
C LVERT IS ZERO THE VERTICAL COMPONENT OF THE FILTER IS THE
C SAME AS THE HORIZONTAL. R(LR) IS THE ARRAY USED TO STORE
C THE LINES BEING PROCESSED.
C
      REAL R(LR)
      INTEGER KL(1)
      COMMON/ERR/JERR,IERR
C
      CALL FLDATA(IA,IU,IATYPE,LL,NL)
      CALL FLSET(IA,0,0,0)
C
C***SET FILTER SIZE
      LFLTH=((MINO(LFLT,LL)+1)/2)*2-1
      LFLTV=LVERT
      IF(LVERT.LE.0) LFLTV=LFLT
      LFLTV=((MINO(LFLTV,NL)+1)/2)*2-1
C***SET FLAGS FOR RELATIVE MODE
      MV=MVERT
      IF(LVERT.LE.0) MV=MFLT
      IF(MV.EQ.2.OR.MV.EQ.4) THEN
        JRELV=1
      ELSE
        JRELV=0
      ENDIF
      IF(MFLT.EQ.2.OR.MFLT.EQ.4) THEN
        JRELH=1
      ELSE
        JRELH=0
      ENDIF
C***SET LENGTH OF OUTPUT LINE
      IF(MFLT.GE.3) THEN
        LLB=LL+(LFLTH-1)
      ELSEIF(MFLT.GE.1) THEN
        LLB=LL
      ELSE
        LLB=LL-(LFLTH-1)
      ENDIF
C***SET LENGTH OF LINES IN BUFFER ARRAY
      IF(JRELH.GT.0) THEN
        LLR=MAXO(LL,LLB+1)
      ELSE
        LLR=MAXO(LL,LLB)
      ENDIF
C***SET SPREADS
      SFLTH=-SFLT
      IF(SFLTH.LT.0.0) SFLTH=-SFLTH*(LL-1)
      SFLTV=-SVERT
      IF(SFLTV.LT.0.0) SFLTV=-SFLTV*(LL-1)
      IF(LVERT.LE.0) SFLTV=SFLTH
C***SET OFFSETS IN BUFFER ARRAY
      KAUX=LLR
      IF(JRELH.GT.0) THEN
        KBUFF=2*LLR
      ELSE
        KBUFF=LLR
      ENDIF
      KFLTV=KBUFF+LFLTV*LLR
      KFLTH=KFLTV+LFLTV+1
```

```
C***SET OUTPUT LINE LIMITS AND THEIR NUMBER
 IF(MV.GE.3) THEN
   NLFIRST=1
   NLLAST=NL+LFLTV-1
   NLB=NLLAST
 ELSEIF(MV.GE.1) THEN
   NLFIRST=1+LFLTV/2
   NLLAST=NL+LFLTV/2
   NLB=NL
 ELSE
   NLFIRST=1+(LFLTV-1)
   NLLAST=NL
   NLB=NL-(LFLTV-1)
 ENDIF
C***SET OFFSETS FOR HORIZONTAL FILTERING
 IF(MFLT.GE.3) THEN
   KLA=1-LFLTH
   KRA=LL-1
   KB=LFLTH
 ELSEIF(MFLT.GE.1) THEN
   KLA=-LFLTH/2
   KRA=LL-1-LFLTH/2
   KB=1+LFLTH/2
 ELSE
   KLA=0
   KRA=LL-LFLTH
   KB=1
 ENDIF
C
C***RETURN IF OUT OF SPACE OR DIMENSIONS NOT POSITIVE
C
 IF((KFLTH+LFLTH).GT.LR.OR.LLB.LE.0.OR.NLB.LE.0) THEN
   IF(IERR.GT.0) WRITE(IERR,*) ' OUT OF SPACE OR PARAMETER ERROR'
   RETURN
 ENDIF
C
 CALL FLSET(-IATYPE,IB,LLB,NLB)
C
C***SET PARAMETERS FOR FIRST FILTER
C
 SF=SFLTH
 LF=LFLTH
 KF=KFLTH
C
C***LOOP OVER HORIZONTAL AND VERTICAL FILTERS
C
 DO 120 J=1,2
C
C  ***GENERATE FILTER
C
   DO 90 I=1,LF
     IF(SF.EQ.0.0) THEN
       R(KF+I)=1.0
     ELSE
       DF=FLOAT(I-(LF+1)/2)/SF
       R(KF+I)=EXP(-0.5*(DF**2))
     ENDIF
   90 CONTINUE
C
C  ***NORMALISE
C
   S=0.0
   DO 100 I=1,LF
   100 S=S+R(KF+I)
   IF(S.NE.0.0) THEN
     DO 110 I=1,LF
     110 R(KF+I)=R(KF+I)/S
   ENDIF
C
C  ***SET PARAMETERS FOR SECOND FILTER
C
   SF=SFLTV
   LF=LFLTV
   KF=KFLTV
C
 120 CONTINUE
C
C***ZERO BUFFER AND SET POINTERS
C
 DO 130 I=1,(LLR*LFLTV)
 130 R(KBUFF+I)=0.0
 LINENO=1
 KNXTL=KBUFF
C
C***LOOP OVER LINES
C
 DO 260 IL=1,NLLAST
C
C  ***ZERO NEXT LINE AND AUX BUFFER IF USED
C
   DO 140 I=1,LLR
   140 R(KNXTL+I)=0.0
   IF(JRELH.GT.0) THEN
     DO 150 I=1,LLR
     150 R(KAUX+I)=0.0
   ENDIF
C
C  ***ENTER NEXT LINE IF DATA, AND FILTER HORIZONTALLY
C
   IF(IL.LE.NL) THEN
     CALL GETLN(IA,R,LL,KL,1.,-1.,K)
     DO 180 IFLT=1,LFLTH
       CFLT=R(KFLTH+IFLT)
       NB=KNXTL+KB-IFLT
       ILA=MAX0(1,KLA+IFLT)
       IRA=MIN0(LL,KRA+IFLT)
       DO 160 IA=ILA,IRA
       160 R(IA+NB)=R(IA+NB)+CFLT*R(IA)
       IF(JRELH.GT.0) THEN
         NAUX=KAUX+KB-IFLT
         DO 170 IA=ILA,IRA
         170 R(IA+NAUX)=R(IA+NAUX)+CFLT
       ENDIF
     180 CONTINUE
     IF(JRELH.GT.0) THEN
       DO 190 I=1,LLB
         IF(R(KAUX+I).NE.0.0) R(KNXTL+I)=R(KNXTL+I)/R(KAUX+I)
       190 CONTINUE
C      ***SET DATA FLAG ON LINE
       R(KNXTL+LLB+1)=1.0
     ENDIF
   ENDIF
C
C  ***SCROLL DOWN VERTICAL FILTER INTO CORRECT POSITION
C
   DO 200 I=1,LFLTV
   200 R(KFLTV+2+LFLTV-I)=R(KFLTV+1+LFLTV-I)
   R(KFLTV+1)=R(KFLTV+LFLTV+1)
C
C  ***OUTPUT LINE IF REQUIRED
C
   IF(IL.GE.NLFIRST.AND.IL.LE.NLLAST) THEN
C
C    ***ZERO OUTPUT LINE
C
     DO 210 I=1,LLB
     210 R(I)=0.0
C
C    ***FILTER VERTICALLY
C
     SC=0.0
     DO 240 I=1,LFLTV
       CFLT=R(KFLTV+I)
       KLN=KBUFF+(I-1)*LLR
       DO 230 IP=1,LLB
       230 R(IP)=R(IP)+CFLT*R(KLN+IP)
C      ***IF RELATIVE ADD COEFFICIENT TO SUM IF LINE WAS DATA
       IF(JRELV.GT.0) THEN
         IF(R(KLN+LLB+1).GT.0.0) SC=SC+CFLT
       ENDIF
     240 CONTINUE
C    ***IF RELATIVE AND NON ZERO SUM, DIVIDE BY SUM
     IF(JRELV.GT.0.AND.SC.NE.0.0) THEN
       DO 250 IP=1,LLB
       250 R(IP)=R(IP)/SC
     ENDIF
C
C    ***WRITE OUT LINE
C
     CALL PUTLN(IB,R,LLB,KL,1.,-1.,K)
C
   ENDIF
C
C  ***ADVANCE POINTER TO NEXT LINE
C
   LINENO=LINENO+1
   IF(LINENO.GT.LFLTV) LINENO=1
   KNXTL=KBUFF+(LINENO-1)*LLR
C
 260 CONTINUE
C
 RETURN
 END
C*** OF FLTXY
```

```
      SUBROUTINE PKFIND (R,LR,IA,IDEL,NMAX)
C
C FINDS THE HIGHEST PEAKS IN THE DATA ON IA, TO A MAXIMUM
C OF NMAX, USING A SPACING PARAMETER IDEL, AND LEAVING THE
C RESULTS IN /PKT/. R(LR) IS TEMPORARY LINE STORAGE.
C
      INTEGER KL(1)
      REAL R(LR)
C
      IF(IA.LE.0) RETURN
      CALL FLDATA(IA,IU,IATYPE,LL,NL)
      CALL FLSET(IA,0,0,0)
C
      ID=MAXO(0,IDEL)
      CALL PKINIT(NMAX,ID,LL,NL,IA)
C
      DO 110 IL=1,NL
        CALL GETLN(IA,R,LL,KL,1,-1,K)
        JUP=0
        RMAX=1.0E20
C
        DO 100 I=1,LL
C         ***TRACK MAX DATA VALUE
          IF(R(I).GT.RMAX) THEN
            RMAX=R(I)
            IMAX=I
            JUP=1
          ELSEIF(JUP.LE.0) THEN
            RMAX=R(I)
            IMAX=I
          ENDIF
C         ***PROCESS A PEAK
          IF(I.GT.(IMAX+ID)) THEN
            CALL PKPRO(RMAX,IMAX,IL)
            RMAX=R(I)
            IMAX=I
            JUP=0
          ENDIF
        100 CONTINUE
C       ***PROCESS ANY RESIDUAL PEAK ON LINE
        IF(JUP.GT.0) CALL PKPRO(RMAX,IMAX,IL)
C       ***REMOVE GARBAGE PEAKS
        CALL PKSWP(IL)
      110 CONTINUE
C
C***REMOVE GARBAGE PEAKS
      CALL PKSWP(0)
C
      RETURN
      END
C*** OF PKFIND
C
C
C
      SUBROUTINE PKINIT (NMX,IDEL,L,N,IA)
C
C INITIALISES /PKT/ BEFORE FINDING PEAKS IN ARRAY OF DATA.
C
      COMMON/PKT/NMAX,ID,IDSQ,LL,NL,NPK,IU,IPK(50),LPK(50),P(50)
C
      NMAX=MINO(50,NMX)
      IF(NMAX.LE.0) NMAX=20
      ID=IDEL
      IDSQ=(IDEL**2)+2
      LL=L
      NL=N
      NPK=0
      IU=IA
C
      RETURN
      END
C*** OF PKINIT
C
C
C
      SUBROUTINE PKPRO (RMAX,IMAX,IL)
C
C UPDATES /PKT/ USING THE PEAK GIVEN AS PARAMETERS.
C
      COMMON/PKT/NMAX,ID,IDSQ,LL,NL,NPK,IU,IPK(50),LPK(50),P(50)
C
C***RETURN IF TABLE FULL AND NEW PEAK LESS THAN LAST IN TABLE
      IF(NPK.GE.NMAX) THEN
        IF(RMAX.LT.P(NMAX)) RETURN
      ENDIF
C
      JIN=0
C
C***SCAN PEAK TABLE IN DIRECTION OF DECREASING PEAK VALUE
C
      IF(NPK.GT.0) THEN
        I=0
        100 I=I+1
C         ***TEST IF NEW PEAK IS CLOSE TO EXISTING PEAK
          IF((IL-IABS(LPK(I))).LT.ID) THEN
            IF(((IMAX-IPK(I))**2+(IL-IABS(LPK(I)))**2).LE.IDSQ) THEN
C
              IF(P(I).GT.RMAX) THEN
C               ***PART OF EXISTING PEAK, NO EFFECT UNLESS LPK FLAGGED
                IF(LPK(I).LT.0) THEN
                  CALL PKDEL(I)
                  CALL PKIN(RMAX,IMAX,-IL)
                ENDIF
                RETURN
              ELSEIF(JIN.LE.0) THEN
C               ***NEW PARAMETERS TO EXISTING PEAK
                CALL PKDEL(I)
                CALL PKIN(RMAX,IMAX,IL)
C               ***FLAG AS ENTERED IN TABLE
                JIN=1
              ELSE
C               ***WEAKER PEAK MERGED INTO UPDATED STRONGER
                CALL PKDEL(I)
                I=I-1
              ENDIF
C
            ENDIF
          ENDIF
        IF(I.LT.NPK) GO TO 100
      ENDIF
C
C***ISOLATED PEAK, ENTER IN TABLE
C
      IF(JIN.LE.0) CALL PKIN(RMAX,IMAX,-IL)
C
      RETURN
      END
C*** OF PKPRO
C
C
C
      SUBROUTINE PKIN (RMAX,IMAX,IL)
C
C INSERTS PEAK GIVEN AS PARAMETERS IN ORDERED TABLE /PKT/.
C
      COMMON/PKT/NMAX,ID,IDSQ,LL,NL,NPK,IU,IPK(50),LPK(50),P(50)
C
C***SCAN TABLE TO FIND FIRST ENTRY LOWER THAN RMAX AND SO
C***LOCATE POSITION FOR NEW ENTRY
C
      IP=0
      100 IP=IP+1
        IF(IP.GT.NPK) THEN
          J=1
        ELSEIF(RMAX.GE.P(IP)) THEN
          J=1
        ELSE
          J=0
        ENDIF
      IF(J.LE.0) GO TO 100
C
C***UPDATE POINTER TO LAST ENTRY
C
      NPK=MINO(NMAX,(NPK+1))
C
C***IF NEW ENTRY WILL PRECEDE LAST, MOVE DOWN THOSE TO FOLLOW
C
      IF(IP.LT.NPK) THEN
        I=NPK
        110 I=I-1
          IPK(I+1)=IPK(I)
          LPK(I+1)=LPK(I)
          P(I+1)=P(I)
        IF(I.GT.IP) GO TO 110
      ENDIF
C
C***ENTER NEW PEAK IF WITHIN BOUNDS OF TABLE
C
      IF(IP.LE.NMAX) THEN
        IPK(IP)=IMAX
        LPK(IP)=IL
        P(IP)=RMAX
      ENDIF
C
      RETURN
      END
C*** OF PKIN
C
C
C
      SUBROUTINE PKDEL (I)
C
C DELETES PEAK I FROM /PKT/.
C
      COMMON/PKT/NMAX,ID,IDSQ,LL,NL,NPK,IU,IPK(50),LPK(50),P(50)
C
      NPK=NPK-1
      IF(NPK.LE.0) RETURN
      DO 100 J=I,NPK
        IPK(J)=IPK(J+1)
        LPK(J)=LPK(J+1)
        P(J)=P(J+1)
      100 CONTINUE
C
      RETURN
      END
C*** OF PKDEL
```

```
      SUBROUTINE PKSWP (IL)
C
C SWEEPS UP ALL FLAGGED PEAKS NOW OUT OF RANGE
C
      COMMON/PKT/NMAX,ID,IDSQ,LL,NL,NPK,IU,IPK(50),LPK(50),P(50)
C
      IF(NPK.LE.0) RETURN
      I=NPK+1
  100 I=I-1
        IF(LPK(I).LT.0.AND.(IL.LE.0.OR.(IL+LPK(I)).GT.ID))
     +    CALL PKDEL(I)
      IF(I.GT.1) GO TO 100
C
      RETURN
      END
C*** OF PKSWP
C
C
C
      SUBROUTINE PKLST (IOUT,IBOR)
C
C LISTS THE TABLE OF PEAKS IN /PKT/ TO UNIT IOUT. ZERO
C DEFAULTS TO 6 AND OUTPUTS THE INTEGER POSITIONS ONLY. IF
C IBOR IS NOT ZERO, RELATIVE POSITIONS WITHIN THE WINDOW ARE
C ALSO OUTPUT AND THOSE PEAKS WITHIN ARE SERIALLY NUMBERED.
C
      COMMON/PKT/NMAX,ID,IDSQ,LL,NL,NPK,IU,IPK(50),LPK(50),P(50)
C
      IU=IOUT
      IF(IU.LE.0) IU=6
      CREL=FLOAT(LL-2*IBOR-1)
      INUM=0
C
      DO 110 I=1,NPK
        IX=LPK(I)
        IY=IPK(I)
        PK=P(I)
        X=FLOAT(IX-1)/FLOAT(LL-1)
        Y=FLOAT(IY-1)/FLOAT(LL-1)
C
        IF(IOUT.LE.0.OR.IX.LE.0) THEN
          WRITE(IU,1100) I,IX,IY,PK
        ELSEIF(IBOR.EQ.0.OR.CREL.LE.0.0) THEN
          WRITE(IU,1100) I,IX,IY,PK,X,Y
        ELSE
          XREL=FLOAT(IX-1-IBOR)/CREL
          YREL=FLOAT(IY-1-IBOR)/CREL
          IF(IX.LE.(NL-IBOR).AND.IX.GT.IBOR.AND.
     +       IY.LE.(LL-IBOR).AND.IY.GT.IBOR) THEN
            INUM=INUM+1
            WRITE(IU,1100) I,IX,IY,PK,X,Y,XREL,YREL,INUM
          ELSE
            WRITE(IU,1100) I,IX,IY,PK,X,Y,XREL,YREL
          ENDIF
        ENDIF
        1100 FORMAT(1X,I3,1H/,2I4,F10.4,4X,2F7.4,4X,2F7.4,I5)
  110 CONTINUE
C
      WRITE(IU,1120) NMAX,NL,LL,IBOR
      1120 FORMAT(/' MAX PEAKS=',I3,'  NL=',I3,'  LL=',I3,
     +  '  IBOR=',I3)
C
      RETURN
      END
C*** OF PKLST
C
C
C
      SUBROUTINE PKCEN (R,LR,IA,IB,IBOR)
C
C LOCATES THE CENTRES OF THE PEAKS ALREADY FOUND BY
C PKFIND AND LISTS THEM TO IB. IF IA.LE.0 EXISTING
C DATA ONLY IS OUTPUT. IF IB IS 0 OUTPUT DEFAULTS TO
C 6. IF IB IS -VE THERE IS NO OUTPUT AND RESULTS ARE
C LEFT IN COMMON /PKC/.
C
      REAL R(LR)
      INTEGER KT(1)
      COMMON/PKT/NMAX,ID,IDSQ,LL,NL,NPK,IU,IPK(50),LPK(50),P(50)
      COMMON/PKC/JB(50),PL(50),PR(50),PT(50),PB(50)
C
C***EXTRACT REQUIRED DATA FROM FILE
C
      IF(IA.GT.0) THEN
        CALL FLDATA(IA,IU,IATYPE,LLA,NLA)
        CALL FLSET(IA,0,0,0)
C  ***FLAG THOSE PEAKS NOT ON EDGE
        DO 100 I=1,NPK
          JB(I)=0
          IF(IPK(I).GT.1.AND.IPK(I).LT.LLA.AND.
     +       LPK(I).GT.1.AND.LPK(I).LT.NLA) JB(I)=1
  100   CONTINUE
C
        DO 120 IL=1,NLA
          CALL GETLN(IA,R,LLA,KT,1,-1,K)
          DO 110 I=1,NPK
            IF(JB(I).GE.1) THEN
              IF(IL.EQ.LPK(I)-1) THEN
                PT(I)=R(IPK(I))
              ELSEIF(IL.EQ.LPK(I)+1) THEN
                PB(I)=R(IPK(I))
              ELSEIF(IL.EQ.LPK(I)) THEN
                PL(I)=R(IPK(I)-1)
                PR(I)=R(IPK(I)+1)
              ENDIF
            ENDIF
  110     CONTINUE
  120   CONTINUE
      ENDIF
C
C***INTERPOLATE AND OUTPUT RESULT
C
      IF(IB.GE.0) THEN
        IUB=IB
        IF(IUB.EQ.0) IUB=6
        WRITE(IUB,*) ' INTERPOLATED PEAK POSITIONS'
        XMAX=FLOAT(NL-1-2*IBOR)/FLOAT(LL-1-2*IBOR)
        DO 130 I=1,NPK
          IF(JB(I).GE.1) THEN
C
            A=0.5*PT(I)-P(I)+0.5*PB(I)
            B=0.5*(PB(I)-PT(I))
            C=P(I)
            PX=-B/(2.0*A)
            FACTV=1.0-(B**2)/(4.0*A*C)
C
            A=0.5*PL(I)-P(I)+0.5*PR(I)
            B=0.5*(PR(I)-PL(I))
            C=P(I)
            PY=-B/(2.0*A)
            FACTH=1.0-(B**2)/(4.0*A*C)
C
            PC=P(I)*FACTH*FACTV
C
            X=(FLOAT(LPK(I)-IBOR)+PX-1.0)/FLOAT(LL-1-2*IBOR)
            Y=(FLOAT(IPK(I)-IBOR)+PY-1.0)/FLOAT(LL-1-2*IBOR)
C
            IF(X.GE.0.0.AND.X.LE.XMAX.AND.Y.GE.0.0.AND.Y.LE.1.0) THEN
              WRITE(IUB,1120) I,PC,X,Y
              1120 FORMAT(1X,I3,1H/,F10.4,' ....',10F7.4)
            ELSE
              WRITE(IUB,1125) I,PC,X,Y
              1125 FORMAT(1X,I3,1H/,F10.4,'     ',10F7.4)
            ENDIF
C
          ENDIF
  130   CONTINUE
      ENDIF
C
      RETURN
      END
C*** OF PKCEN
C
C
C
      SUBROUTINE PEAKS (R,LR,IA,IDEL,NMAX,IBOR,IB,IC)
C
C FINDS PEAKS AND ESTIMATES CENTRES FROM IA. IF IA -VE IT
C OUTPUTS OLD DATA. IDEL, NMAX, CONTROL FINDING PEAKS,
C IBOR THE BORDER ON OUTPUT. IB SELECTS OUTPUT UNIT, 0
C DEFAULTING TO 6, AND -VE WITH NO OUTPUT OF CENTRE
C ESTIMATES. FINDING TABLE IS OUTPUT TO 6 FOR IB +6 OR -6.
C IF IC +VE FULL PRINTER FORMATTED OUTPUT IS PRODUCED TO IC.
C
      REAL R(LR)
C
      CALL PKFIND(R,LR,IA,IDEL,NMAX)
      IF(IB.EQ.6.OR.IB.EQ.-6) CALL PKLST(6,IBOR)
      CALL PKCEN(R,LR,IA,IB,IBOR)
C
      IF(IC.GT.0) THEN
        WRITE(IC,1000)
        1000 FORMAT(' ',///)
        CALL PKLST(IC,IBOR)
        WRITE(IC,1010)
        1010 FORMAT(' ',///)
        CALL PKCEN(R,LR,0,IC,IBOR)
        WRITE(IC,1010)
      ENDIF
C
      RETURN
      END
C*** OF PEAKS
```

```
      SUBROUTINE FLSUM (R,LR,S,LS,IA,IB,IC,SCALEA,SCALEB,ADDC,ITRIM)
C
C SUMS THE ARRAYS IN IA, IB IN THE AREA OF THEIR OVERLAP WITH
C WEIGHTS SCALEA, SCALEB AND A CONSTANT TERM ADDC, AND WRITES
C THE RESULT TO IC, AFTER TRIMMING OFF A BORDER ITRIM ON ALL
C SIDES. IA OR IB CAN BE ZERO.
C
      COMMON/ERR/JERR,IERR
      REAL R(LR),S(LS)
      INTEGER KT(1)
C
      ITRM=MAX0(0,ITRIM)
      CALL FLDATA(IA,IUA,IATYPE,LLA,NLA)
      CALL FLDATA(IB,IUB,IBTYPE,LLB,NLB)
      IF(IA.LE.0) THEN
        LLCOM=LLB
        NLCOM=NLB
      ELSEIF(IB.LE.0) THEN
        LLCOM=LLA
        NLCOM=NLA
      ELSE
        LLCOM=MINO(LLA,LLB)
        NLCOM=MINO(NLA,NLB)
      ENDIF
C***CALC OUTPUT ARRAY SIZE
      LLC=LLCOM-2*ITRM
      NLC=NLCOM-2*ITRM
C
      IF(IC.LE.0.OR.(IA.LE.0.AND.IB.LE.0).OR.
     +   IA.EQ.IB.OR.((IA.EQ.IC.OR.IB.EQ.IC).AND.IC.NE.0).OR.
     +   LLC.LE.0.OR.NLC.LE.0.OR.
     +   LLA.GT.LR.OR.LLB.GT.LR.OR.(LLC+ITRM).GT.LS) THEN
        IF(IERR.GT.0) WRITE(IERR,*) ' PARAMETER ERROR'
        RETURN
      ENDIF
C
C***REWIND INPUT AND INITIALISE OUTPUT FILES
      CALL FLSET(IA,0,0,0)
      CALL FLSET(IB,0,0,0)
      CALL FLSET(0,IC,LLC,NLC)
C
C***LOOP OVER LINES
C
      DO 140 IL=1,(NLC+ITRM)
C
        DO 100 I=1,(LLC+ITRM)
        100 S(I)=ADDC
C
        IF(IA.GT.1) THEN
          CALL GETLN(IA,R,LLA,KT,1,-1,K)
          DO 110 I=1,LLC+ITRM
          110 S(I)=S(I)+R(I)*SCALEA
        ENDIF
C
        IF(IB.GT.1) THEN
          CALL GETLN(IB,R,LLB,KT,1,-1,K)
          DO 120 I=1,LLC+ITRM
          120 S(I)=S(I)+R(I)*SCALEB
        ENDIF
C
        IF(ITRM.GT.0) THEN
          DO 130 I=1,LLC
          130 S(I)=S(I+ITRM)
        ENDIF
C
        IF(IC.GT.1) THEN
          IF(IL.GT.ITRM) CALL PUTLN(IC,S,LLC,KT,1,-1,K)
        ENDIF
C
      140 CONTINUE
C
      RETURN
      END
C*** OF FLSUM
C
C
C
      SUBROUTINE FLLST (IA,IB,MPTS,MLINES,SCALE)
C
C LISTS THE FILE OF DATA IA TO IB AS A PRINTABLE
C ARRAY OF SMALL INTEGERS. MPTS,MLINES SET THE
C MAX NO OF PTS ON A LINE AND LINES OUTPUT, .LE.0
C SELECTING DEFAULTS 20,28 AND CAUSING -MPTS,
C -MLINES POINTS, LINES TO BE SKIPPED. IB 0
C DEFAULTS TO 6, IB -VE SELECTS -IB BUT WITH
C PRINTER FORMAT AND DEFAULT PARAMETERS 40,80.
C SCALE IS THE DATA SCALING FACTOR AND IF 0 IS
C SET AUTOMATICALLY FROM THE ENTIRE FILE OF DATA.
C
      INTEGER IDATA(300)
      REAL DATA(300)
C
      CALL FLDATA(IA,IU,IATYPE,LL,NL)
      IF(IU.LE.0.OR.LL.LE.0.OR.LL.GT.300.OR.NL.LE.0) RETURN
      IUB=IABS(IB)
      IF(IB.EQ.0) IUB=6
C
      LPL=1
      LPR=20
      IF(IB.LT.0) LPR=40
      IF(MPTS.GT.0) THEN
        LPR=MINO(LL,MPTS)
      ELSE
        LPL=MINO(LL,LPL-MPTS)
        LPR=MINO(LL,LPR-MPTS)
      ENDIF
C
      NLT=1
      NLB=28
      IF(IB.LT.0) NLB=80
      IF(MLINES.GT.0) THEN
        NLB=MINO(NL,MLINES)
      ELSE
        NLT=MINO(NL,NLT-MLINES)
        NLB=MINO(NL,NLB-MLINES)
      ENDIF
C
C***SET SCALING FACTOR
      C=SCALE
      IF(C.EQ.0.0) THEN
        REWIND IA
        70 CONTINUE
          READ(IA,END=90) (DATA(IP),IP=1,LL)
          DO 80 IP=1,LL
          80 C=AMAX1(C,ABS(DATA(IP)))
        GO TO 70
        90 CONTINUE
        C=99.0/AMAX1(C,0.0001)
      ENDIF
C
      REWIND IA
      WRITE(IUB,1095) LPL,NL,LL
      1095 FORMAT('1 FIRST COLUMN ',I3,'   SIZE IS ',
     +  I3,' LINES OF ',I3,' POINTS')
      IL=0
C
C***LOOP OVER LINES
C
      100 CONTINUE
        IL=IL+1
        READ(IA,END=140) (DATA(IP),IP=1,LL)
        IF(IL.GE.NLT) THEN
          DO 110 IP=1,LL
          110 IDATA(IP)=IFIX(C*DATA(IP))
          IF(IB.GE.0) THEN
            IF((LPR-LPL).LE.19) THEN
              WRITE(IUB,1120) IL,(IDATA(IP),IP=LPL,LPR)
              1120 FORMAT(1X,I3,1H/,20I3)
            ELSE
              WRITE(IUB,1130) IL,(IDATA(IP),IP=LPL,LPR)
              1130 FORMAT(1X,I3,1H/,20I3,14(/5X,20I3))
            ENDIF
          ELSE
            IF((LPR-LPL).LE.39) THEN
              WRITE(IUB,1140) IL,(IDATA(IP),IP=LPL,LPR)
              1140 FORMAT(1H0,I3,1H/,40I3)
            ELSE
              WRITE(IUB,1150) IL,(IDATA(IP),IP=LPL,LPR)
              1150 FORMAT(1H0,I3,1H/,40I3,5(/5X,40I3))
            ENDIF
          ENDIF
        ENDIF
      IF(IL.LT.NLB) GO TO 100
C
      140 RETURN
      END
C*** OF FLLST
C
C
C
      SUBROUTINE FLSTAT (R,LR,IA,BASE,RANGE,IUB,IC,BLOG)
C
C ANALYSES THE DISTRIBUTION OF DATA VALUES ON FILE
C IA AND OUTPUTS STATISTICS AND A DISTRIBUTION TO IUB AND
C IC IF NOT ZERO. DISTRIBUTION COVERS VALUES IN GIVEN RANGE
C ABOVE BASE. IF IUB IS 0, OUTPUT IS A SHORT FORM TO
C UNIT 6. IF -VE THERE IS NO OUTPUT BUT THE RESULTS ARE
C LEFT IN COMMON /HST/. IF IA IS 0 THE DATA IN /HST/ WILL
C BE OUTPUT. IF BASE AND RANGE ARE BOTH 0 ONLY THE
C STATISTICS ARE PRODUCED. IF RANGE ONLY IS 0 THE
C DISTRIBUTION HAS LOG INTERVALS. R(LR) IS THE LINE BUFFER.
C
      REAL R(LR)
      INTEGER KT(1)
      COMMON/HST/BS,RNG,DBS,T,SQ,H,B,LL,NL,NH(22)
C
C***INITIALISE
C
      IF(IA.GT.0) THEN
        CALL FLDATA(IA,IU,IATYPE,LL,NL)
        CALL FLSET(IA,0,0,0)
        BS=BASE
        RNG=RANGE
        DBS=BLOG
        T=0.0
        SQ=0.0
        DO 100 I=1,22
        100 NH(I)=0.0
```

```
      H=-1.0E20
      B=-H
    ENDIF
C
    IB=IUB
    IF(IB.EQ.0) IB=6
C
    IF(BS.EQ.0.0.AND.RNG.EQ.0.0) THEN
      JHIST=0
    ELSEIF(RNG.EQ.0.0) THEN
      JHIST=1
      JLOG=1
      BIN=0.2
      S=4.2
    ELSE
      JHIST=1
      JLOG=0
      BIN=RNG/20.0
      S=BS+RNG+BIN
    ENDIF
C
C***ANALYSE
C
    IF(IA.GT.0) THEN
      DO 120 IL=1,NL
        CALL GETLN(IA,R,LL,KT,1,-1,K)
C
        DO 110 IP=1,LL
          D=R(IP)
          T=T+D
          SQ=SQ+D**2
          H=AMAX1(H,D)
          B=AMIN1(B,D)
C
          IF(JHIST.GE.1) THEN
            IF(JLOG.GE.1) THEN
              DJ=ALOG10(AMAX1((D-DBS),1.0E-20)/BS)
              J=INT((S-DJ)/BIN)+1
            ELSE
              J=INT((S-D)/BIN)+1
            ENDIF
            IF(J.GT.21) THEN
              NH(22)=NH(22)+1
            ELSEIF(J.LE.1) THEN
              NH(1)=NH(1)+1
            ELSE
              NH(J)=NH(J)+1
            ENDIF
          ENDIF
        110 CONTINUE
      120 CONTINUE
    ENDIF
C
C***OUTPUT
C
    IF(JHIST.GE.1) THEN
      IF(IC.GT.0) WRITE(IC,1200)
      IF(JLOG.GE.1) THEN
        IF(IB.GT.0) WRITE(IB,1130) BS,DBS
        IF(IC.GT.0) WRITE(IC,1130) BS,DBS
        1130 FORMAT(' BIN BASE,LOG PERCENT  CUM PC     REF='
+       F10.4,'  ABOVE',F10.4)
      ELSE
        IF(IB.GT.0) WRITE(IB,1140)
        IF(IC.GT.0) WRITE(IC,1140)
        1140 FORMAT(' BIN    BASE  PERCENT  CUM PC')
      ENDIF
C
      CUM=0.0
      DO 1160 I=1,22
        PC=FLOAT(NH(I)*100)/FLOAT(NL*LL)
        CUM=CUM+PC
        BOTTOM=S-BIN*FLOAT(I)
        IF(I.LT.22) THEN
          IF(IB.GT.0) WRITE(IB,1150) I-1,BOTTOM,PC,CUM
          IF(IC.GT.0) WRITE(IC,1150) I-1,BOTTOM,PC,CUM
          1150 FORMAT(1X,I2,F10.4,2F8.2)
        ELSE
          IF(IB.GT.0) WRITE(IB,1155) I-1,PC,CUM
          IF(IC.GT.0) WRITE(IC,1155) I-1,PC,CUM
          1155 FORMAT(1X,I2,10X,2F8.2)
        ENDIF
      1160 CONTINUE
    ENDIF
C
    AVG=T/FLOAT(NL*LL)
    STD=SQRT(SQ/FLOAT(NL*LL)-AVG**2)
    IF(IC.GT.0) WRITE(IC,1200)
    IF(IB.GT.0) WRITE(IB,1170) AVG,STD,H,B
    IF(IC.GT.0) WRITE(IC,1170) AVG,STD,H,B
    1170 FORMAT(' AVG=',F10.4,',  SIGMA=',F10.4,',  MAX=',
+   F10.4,',  MIN=',F10.4)
    IF(IC.GT.0) WRITE(IC,1200)
C
    1200 FORMAT(' '//)
C
    RETURN
    END
C*** OF FLSTAT
```

```
    SUBROUTINE FLDATA (IA,IU,IATYPE,LL,NL)
C
C RETURNS THE DIMENSIONS OF IMAGE IN FILE IA.
C
    COMMON/FT/FLL(20),FNL(20)
C
    IU=-1
    IATYPE=0
    LL=0
    NL=0
    IF(IA.GT.0) THEN
      IU=IA
      IATYPE=1
      LL=FLL(IA)
      NL=FNL(IA)
    ENDIF
C
    RETURN
    END
C*** OF FLDATA
C
C
C
    SUBROUTINE FLSET (IA,IB,INCL,INCN)
C
C SETS DIMENSIONS OF FILE IB FROM IA, REWINDS IA AND EMPTIES IB.
C
    COMMON/FT/FLL(20),FNL(20)
C
    LL=INCL
    NL=INCN
    IF(IA.GT.0) THEN
      IF(IA.GT.1) REWIND IA
      LL=FLL(IA)+LL
      NL=FNL(IA)+NL
    ENDIF
    IF(IB.GT.0) THEN
      IF(IB.GT.1) THEN
        REWIND IB
        ENDFILE IB
        REWIND IB
      ENDIF
      FLL(IB)=LL
      FNL(IB)=NL
    ENDIF
C
    RETURN
    END
C*** OF FLSET
C
C
C
    SUBROUTINE FLSIZE (IA,LL,NL)
C
C SETS DIMENSIONS FOR PRE-EXISTING FILE IN /FT/ AND
C REWINDS FILE UNLESS IA -VE.
C
    COMMON/FT/FLL(20),FNL(20)
C
    I=IABS(IA)
    IF(I.GT.0.AND.I.LE.20) THEN
      FLL(I)=LL
      FNL(I)=NL
      IF(IA.GT.0) REWIND IA
    ENDIF
C
    RETURN
    END
C*** OF FLSIZE
```

```
      SUBROUTINE GETLN (IA,DATA,LD,TEXT,LT,J,K)
C
C READS A LINE OF DATA FROM IA IF POSSIBLE, OTHERWISE RETURNS
C ZEROES. IA IS REWOUND AT EOF. TEXT(LT), J, K ARE DUMMIES.
C
      REAL DATA(LD)
      INTEGER TEXT(LT)
C
      K=-1
      IF(IA.GT.1) THEN
        READ(IA,END=100) DATA
        K=0
      ELSE
        DO 90 I=1,LD
        90 DATA(I)=0.0
        K=0
      ENDIF
      TEXT(1)=J
      RETURN
C
      100 REWIND IA
      DO 110 I=1,LD
      110 DATA(I)=0.0
      RETURN
      END
C*** OF GETLN
C
C
C
      SUBROUTINE PUTLN (IA,DATA,LD,TEXT,LT,J,K)
C
C WRITES A LINE OF DATA TO FILE IA. TEXT(LT), J, K ARE DUMMIES.
C
      REAL DATA(LD)
      INTEGER TEXT(LT)
C
      K=-1
      IF(IA.GT.1) THEN
        WRITE(IA) DATA
        K=0
      ENDIF
      TEXT(1)=J
      RETURN
      END
C*** OF PUTLN
```

A3
Programs:
One Dimensional

Routines are listed by name only, and then with brief descriptive notes. The source code follows.

```
SUBROUTINE COPY   (R,LR,LD,NA,NB,AMP,OFFSET,SCALE)
SUBROUTINE MIX    (R,LR,LD,NA,NB,NC,AMPA,AMPB,OFFSET)
SUBROUTINE SCALE  (R,LR,LD,NA,AMP,OFFSET,DIV)
SUBROUTINE CONV   (R,LR,LD,NA,NB,NC,AMP)
SUBROUTINE GAUSS  (R,LR,LD,N,SPREAD,AMP)
SUBROUTINE STAT   (R,LR,LD,NA)
SUBROUTINE NOISE  (R,LR,LD,NA,AMP,Z)
SUBROUTINE MULT   (R,LR,LD,NA,NB,NC,AMP)
SUBROUTINE FNC    (R,LR,LD,NAME,NA,PA,PB,PC,PD,PE,PF)

SUBROUTINE TMIX   (R,LR,LD,NTA,NTB,NTC,AA,AB,OFF)
SUBROUTINE TNOISE (R,LR,LD,NT,AMP,Z)
SUBROUTINE TLINE  (R,LR,LD,NP,NT,AA,XA,AB,XB,OFF)
SUBROUTINE TCONV  (R,LR,LD,NT,NF,NB,AMP,LF)

SUBROUTINE SQ     (R,LR,LD,NA,NB)
SUBROUTINE RD     (R,LR,LD,NA,NB,SCALE,AMP)
SUBROUTINE LN     (R,LR,LD,NA,NB)
SUBROUTINE COPYSQ (R,LR,LD,NA,NB,AMP,OFFSET)
SUBROUTINE MIXSQ  (R,LR,LD,NA,NB,NC,AMPA,AMPB,OFFSET)
SUBROUTINE SCALSQ (R,LR,LD,NA,AMP,OFFSET)
SUBROUTINE CONSQ  (R,LR,LD,NA,NB,NC,AMP)

SUBROUTINE FTRANS (R,LR,LD,NA,NB,SCALE)
SUBROUTINE FTWO   (R,LR,LD,NA,NB,SCAL)
SUBROUTINE SPEC   (R,LR,LD,IU,NA,NB,LL,NL,OFF)
SUBROUTINE FLTGEN (R,LR,LD,NA,LF,IOUT)

SUBROUTINE SAVE   (R,LR,LD,IU,NA,NB)
SUBROUTINE REST   (R,LR,LD,IU,NA,NB)
FUNCTION JV   (LR,LD,NLN,NSQ,NA,NB,NC,ND)
FUNCTION RXT  (R,LR,LD,N,X)
FUNCTION RBIG (R,LR,LD,N)
FUNCTION PSUM (R,LR,LD,NA,NB,IXP)
```

```
SUBROUTINE COPY (R,LR,LD,NA,NB,AMP,OFFSET,SCALE)
  COPIES LINEAR ARRAY NA TO NB, MULTIPLYING BY AMP (0 DEFAULTS
  TO 1) AND ADDING OFFSET. LENGTH IS EXPANDED BY A FACTOR
  SCALE, 0 DEFAULTING TO 1.

SUBROUTINE MIX (R,LR,LD,NA,NB,NC,AMPA,AMPB,OFFSET)
  COPIES THE SUM OF LINEAR ARRAYS NA, NB INTO NC WITH
  WEIGHTING FACTORS AMPA, AMPB AND ADDING OFFSET.

SUBROUTINE SCALE (R,LR,LD,NA,AMP,OFFSET,DIV)
  MULTIPIES A LINEAR ARRAY IN PLACE BY AMP, ADDING OFFSET.
  IF DIV NOT ZERO, FINALLY DIVIDES BY DIV.

SUBROUTINE CONV (R,LR,LD,NA,NB,NC,AMP)
  CONVOLVES TWO LINEAR ARRAYS NA, NB, TAKEN TO BE SYMMETRIC
  ABOUT THEIR FIRST ELEMENTS, TO GIVE A THIRD, NC. IF AMP
  IS NOT ZERO IT IS APPLIED AS A SCALING FACTOR.

SUBROUTINE GAUSS (R,LR,LD,N,SPREAD,AMP)
  ENTERS A GAUSSIAN PROFILE PEAK VALUE AMP (0 DEFAULTS TO 1)
  WIDTH SPREAD IN LINEAR ARRAY N. IF SPREAD IS POSITIVE, IT IS
  RELATIVE TO LD-1, IF NEGATIVE IT IS IN GRID UNITS. IF 0 THE
  FIRST POINT IS SET TO AMP, AND THE REST TO 0.

SUBROUTINE STAT (R,LR,LD,NA)
  DISPLAYS THE MAX, MIN, MEAN AND STD DEVIATION OF LINEAR ARRAY NA.

SUBROUTINE NOISE (R,LR,LD,NA,AMP,Z)
  PUTS GAUSSIAN NOISE IN LINEAR ARRAY NA WITH AMP AS GIVEN
  (0 DEFAULTS TO 1), DERIVED FROM SEED Z (0...1). IF NA
  NEGATIVE, NOISE IS ADDED TO EXISTING CONTENTS. IF Z NEGATIVE
  EXISTING SEED IS USED. SEE IMAGE GENERATION SYSTEM FOR RDG.

SUBROUTINE MULT (R,LR,LD,NA,NB,NC,AMP)
  MULTIPIES LINEAR ARRAYS NA, NB TOGETHER POINT BY POINT
  AND ENTERS INTO ARRAY NC, SCALED BY AMP (0 DEFAULTS TO 1).

SUBROUTINE FNC (R,LR,LD,NAME,NA,PA,PB,PC,PD,PE,PF)
  PERFORMS MATHEMATICAL OPERATIONS POINT BY POINT TO
  GENERATE A LINEAR ARRAY, SPECIFIED BY A CHARACTER
  VALUE NAME AND USING 0,1, OR 2 INPUT LINEAR ARRAYS,
  AND UP TO 6 PARAMETERS.
    0 INPUT ARRAYS ... SINX,COSX,DISC,POLY,QUINT
    1 INPUT ARRAYS ... LOG,ALOG,LN,EXP,SQRT,SQ,RCP,
                       MIN,MAX,CLIP,SIN,COS,DSZ,DS
    2 INPUT ARRAYS ... MLT,DIV
```

```
SUBROUTINE TMIX (R,LR,LD,NTA,NTB,NTC,AA,AB,OFF)
  COPIES THE SUM OF TRIPLE LINEAR ARRAYS NTA,NTB INTO NTC,
  WEIGHTS AA,AB, PLUS AN OFFSET OFF.

SUBROUTINE TNOISE (R,LR,LD,NT,AMP,Z)
  PUTS GAUSSIAN NOISE IN A TRIPLE LINEAR ARRAY NT WITH
  AMPLITUDE AMP (0 DEFAULTS TO 1.0) DERIVED FROM SEED Z (0...1).
  IF NA NEGATIVE NOISE IS ADDED TO EXISTING CONTENTS. IF Z
  NEGATIVE EXISTING SEED IS USED.

SUBROUTINE TLINE (R,LR,LD,NP,NT,AA,XA,AB,XB,OFF)
  PUTS TWO LINES PROFILE NP INTO TRIPLE LINEAR ARRAY NT
  AMPLITUDE AND POSITION AA,XA  AB,XB  PLUS OFFSET OFF.
  POSITION IS MEASURED ALONG CENTRAL LINEAR ARRAY. IF NT
  NEGATIVE OUTPUT ARRAYS ARE NOT INITIALLY ZEROED. IF NP
  NEGATIVE A COMPLETE PROFILE IS ASSUMED RATHER THAN HALF,
  AND POSITION IS TO CENTRE OF NP.

SUBROUTINE TCONV (R,LR,LD,NT,NF,NB,AMP,LF)
  CONVOLVES TRIPLE LINEAR ARRAY NT WITH LINEAR ARRAY NF TO
  FORM LINEAR ARRAY NB. AMP IS A SCALING FACTOR, 0 DEFAULTING
  TO 1. LF LIMITS THE WIDTH OF NF USED, FROM THE CENTRE OF THE
  PROFILE. IF NF NEGATIVE, THE CONVOLVING PROFILE IS CENTRED
  IN NF RATHER THAN BEING SYMMETRIC ABOUT THE ORIGIN.

SUBROUTINE SQ (R,LR,LD,NA,NB)
  FORMS A SQUARE ARRAY NB FROM A LINEAR ARRAY NA BY
  USING IT AS THE X AND Y SEPARABLE COMPONENTS.

SUBROUTINE RD (R,LR,LD,NA,NB,SCALE,AMP)
  FORMS A SQUARE ARRAY NB FROM A LINEAR ARRAY NA BY USING
  IT AS A RADIAL PROFILE, EXPANDING BY A FACTOR SCALE IN
  RADIUS, AND AMP IN AMPLITUDE, 0 DEFAULTING TO 1, AND
  SCALE -1 TO 1.414...

SUBROUTINE LN (R,LR,LD,NA,NB)
  PROJECTS A SQUARE ARRAY NA ONTO A LINEAR ARRAY NB.

SUBROUTINE COPYSQ (R,LR,LD,NA,NB,AMP,OFFSET)
  COPIES SQUARE ARRAY NA TO NB, MULTIPLYING BY AMP (0 DEFAULTS
  TO 1) AND ADDING OFFSET.

SUBROUTINE MIXSQ (R,LR,LD,NA,NB,NC,AMPA,AMPB,OFFSET)
  COPIES THE SUM OF SQUARE ARRAYS NA, NB INTO NC WITH
  WEIGHTING FACTORS AMPA, AMPB (0 DEFAULTS TO 1) AND
  ADDING OFFSET.

SUBROUTINE SCALSQ (R,LR,LD,NA,AMP,OFFSET)
  MULTIPLIES A SQUARE ARRAY IN PLACE BY AMP, ADDING OFFSET.

SUBROUTINE CONSQ (R,LR,LD,NA,NB,NC,AMP)
  FORMS A LINEAR ARRAY NC BY CONVOLVING SQUARE ARRAYS
  NA, NB. IF AMP IS NOT ZERO IT IS APPLIED AS A SCALING
  FACTOR.
```

```
SUBROUTINE FTRANS (R,LR,LD,NA,NB,SCALE)
  GENERATES THE DISCRETE COSINE TRANSFORM OF LINEAR ARRAY NA
  INTO NB. SCALE IF GREATER THAN 1 CAUSES THE PERIOD USED
  TO BE GREATER THAN 2*LD-1, DATA BEING ASSUMED 0 OUTSIDE
  RANGE GIVEN, AND THE HIGHER FREQUENCIES OF THE TRANSFORM
  NOT BEING CALCULATED.

SUBROUTINE FTWO (R,LR,LD,NA,NB,SCAL)
  COSINE TRANSFORMS LINEAR ARRAY NA TO NB, INTERPRETED AS
  A TWO-D TRANSFORM OF A CIRCULAR PROFILE. SCAL GREATER
  THAN 1 EXPANDS IN OUTPUT SPACE.

SUBROUTINE SPEC (R,LR,LD,IU,NA,NB,LL,NL,OFF)
  GENERATES A SPECTRUM IMAGE ON FILE IU, LINE LENGTH LL,
  NUMBER OF LINES NL, BY MULTIPLYING A SPECTRUM PROFILE IN
  LINEAR ARRAY NA BY A TRANSVERSE PROFILE IN NB, BOTH SCALED
  TO REQUIRED LENGTH, AND ADDING AN OFFSET VALUE OFF.

SUBROUTINE FLTGEN (R,LR,LD,NA,LF,IOUT)
  WRITES A FILTER ARRAY OF LF BY LF POINTS TO UNIT
  IOUT, FORMED FROM SQUARE ARRAY NA. LF SHOULD BE ODD.

SUBROUTINE SAVE (R,LR,LD,IU,NA,NB)
  WRITES FROM BUFFER TO UNIT IU. IT STARTS WITH LINEAR ARRAY
  NA AND ENDS WITH NB. IF NB IS LESS THAN NA ONLY NA IS WRITEN.
  IF NA AND NB ARE BOTH ZERO THE ENTIRE USABLE BUFFER IS DUMPED.

SUBROUTINE REST (R,LR,LD,IU,NA,NB)
  READS INTO BUFFER FROM IU, INTO LINEAR ARRAYS FROM NA TO NB.
  IF NB IS LESS THAN NA, INTO NA ONLY. IF NA AND NB ARE BOTH
  ZERO, FROM THE START TO END OF DATA OR BUFFER.

FUNCTION JV (LR,LD,NLN,NSQ,NA,NB,NC,ND)
  RETURNS 1 IF ARRAYS NUMBERS NA...ND ARE VALID, THE
  NUMBERS THAT ARE TO BE LINEAR AND SQUARE BEING GIVEN
  AS NLN, NSQ, RETURNS 0 OTHERWISE AND WRITES A MESSAGE.

FUNCTION RXT (R,LR,LD,N,X)
  RETURNS A LINEAR INTERPOLATED VALUE FOR POSITION ABS(X)
  IN ARRAY N TAKEN AS LINEAR, X BEING IN GRID UNITS. IT
  RETURNS 0 IF OVERRANGE.

FUNCTION RBIG (R,LR,LD,N)
  RETURNS THE LARGEST ABSOLUTE VALUE IN LINEAR ARRAY N.

FUNCTION PSUM (R,LR,LD,NA,NB,IXP)
  SUMS LINEAR ARRAY NA OVER ITS ELEMENTS, WEIGHTED BY LINEAR
  ARRAY NB IF NB POSITIVE, AND BY THE POSITION TO THE POWER IXP.
  IF IXP IS NEGATIVE THE POWER IS -IXP-1 AND THE LAST POINT
  IS WEIGHTED 0.5.
```

```
 SUBROUTINE COPY (R,LR,LD,NA,NB,AMP,OFFSET,SCALE)
C
C COPIES LINEAR ARRAY NA TO NB, MULTIPLYING BY AMP (0 DEFAULTS
C TO 1) AND ADDING OFFSET. LENGTH IS EXPANDED BY A FACTOR
C SCALE, 0 DEFAULTING TO 1.
C
 REAL R(LR)
C
 IF(JV(LR,LD,2,0,NA,NB,0,0).NE.1) RETURN
C
 A=AMP
 IF(A.EQ.0.0) A=1.0
 KB=NB*LD
C
 S=SCALE
 IF(S.LE.0.0) S=1.0
 DO 100 I=0,LD-1
 100 R(KB+I)=A*RXT(R,LR,LD,NA,FLOAT(I)/S)+OFFSET
C
 RETURN
 END
C*** OF COPY
C
C
C
 SUBROUTINE MIX (R,LR,LD,NA,NB,NC,AMPA,AMPB,OFFSET)
C
C COPIES THE SUM OF LINEAR ARRAYS NA, NB INTO NC WITH
C WEIGHTING FACTORS AMPA, AMPB AND ADDING OFFSET.
C
 REAL R(LR)
C
 IF(JV(LR,LD,3,0,NA,NB,NC,0).NE.1) RETURN
C
 AA=AMPA
 AB=AMPB
 KA=NA*LD
 KB=NB*LD
 KC=NC*LD
C
 DO 100 I=0,LD-1
 100 R(KC+I)=AA*R(KA+I)+AB*R(KB+I)+OFFSET
C
 RETURN
 END
C*** OF MIX
C
C
C
 SUBROUTINE SCALE (R,LR,LD,NA,AMP,OFFSET,DIV)
C
C MULTIPIES A LINEAR ARRAY IN PLACE BY AMP, ADDING OFFSET.
C IF DIV NOT ZERO, FINALLY DIVIDES BY DIV.
C
 REAL R(LR)
C
 IF(JV(LR,LD,1,0,NA,0,0,0).NE.1) RETURN
C
 KA=NA*LD
 DO 100 I=0,LD-1
 100 R(KA+I)=AMP*R(KA+I)+OFFSET
C
 IF(DIV.NE.0.0) THEN
   DO 110 I=0,LD-1
   110 R(KA+I)=R(KA+I)/DIV
 ENDIF
C
 RETURN
 END
C*** OF SCALE
C
C
C
 SUBROUTINE CONV (R,LR,LD,NA,NB,NC,AMP)
C
C CONVOLVES TWO LINEAR ARRAYS NA, NB, TAKEN TO BE SYMMETRIC
C ABOUT THEIR FIRST ELEMENTS, TO GIVE A THIRD, NC. IF AMP
C IS NOT ZERO IT IS APPLIED AS A SCALING FACTOR.
C
 REAL R(LR)
C
 IF(JV(LR,LD,3,0,NA,NB,NC,0).NE.1) RETURN
C
 A=1.0/SQRT(FLOAT(2*LD-1))
 IF(AMP.NE.0.0) A=AMP
 KA=NA*LD
 KB=NB*LD
 KC=NC*LD
C
 DO 130 IC=0,LD-1
   ILAP=LD-1-IC
   S=0.0
C
   DO 100 IA=0,IC
     IB=IC-IA
     S=S+R(KA+IA)*R(KB+IB)
   100 CONTINUE
C
   IF(ILAP.GT.0) THEN
     DO 110 IA=IC+1,LD-1
       IB=IA-IC
       S=S+R(KA+IA)*R(KB+IB)
     110 CONTINUE
C
     DO 120 IA=1,ILAP
       IB=IA+IC
       S=S+R(KA+IA)*R(KB+IB)
     120 CONTINUE
   ENDIF
C
   R(KC+IC)=S*A
 130 CONTINUE
C
 RETURN
 END
C*** OF CONV
C
C
C
 SUBROUTINE GAUSS (R,LR,LD,N,SPREAD,AMP)
C
C ENTERS A GAUSSIAN PROFILE PEAK VALUE AMP (0 DEFAULTS TO 1)
C WIDTH SPREAD IN LINEAR ARRAY N. IF SPREAD IS +VE, IT IS
C RELATIVE TO LD-1, IF -VE IT IS IN GRID UNITS. IF 0 THE FIRST
C POINT IS SET TO AMP, AND THE REST TO 0.
C
 REAL R(LR)
C
 IF(JV(LR,LD,1,0,N,0,0,0).NE.1) RETURN
C
 KA=N*LD
 S=SPREAD*FLOAT(LD-1)
 IF(SPREAD.LT.0.0) S=-SPREAD
 A=AMP
 IF(A.EQ.0.0) A=1.0
C
 IF(S.EQ.0.0) THEN
   R(KA)=A
   DO 100 I=1,LD-1
   100 R(KA+I)=0.0
 ELSE
   DO 110 I=0,LD-1
   110 R(KA+I)=A*EXP(-0.5*(FLOAT(I)/S)**2)
 ENDIF
C
 RETURN
 END
C*** OF GAUSS
C
C
C
 SUBROUTINE STAT (R,LR,LD,NA)
C
C DISPLAYS THE MAX, MIN, MEAN AND STD DEVIATION OF LINEAR ARRAY NA.
C
 REAL R(LR)
 COMMON/SYSVAR/IUMSG,ISYS(15)
C
 IF(JV(LR,LD,1,0,NA,0,0,0).NE.1.OR.IUMSG.LE.0) RETURN
C
 RMX=-1.0E20
 RMN=-RMX
 S=0.0
 SSQ=0.0
C
 DO 100 I=NA*LD,NA*LD+LD-1
   RMX=AMAX1(RMX,R(I))
   RMN=AMIN1(RMN,R(I))
   S=S+R(I)
   SSQ=SSQ+R(I)**2
 100 CONTINUE
C
 RAVG=S/FLOAT(LD)
 SD=SQRT(SSQ/FLOAT(LD)-RAVG**2)
 WRITE(IUMSG,1110) RMN,RMX,RAVG,SD
 1110 FORMAT(1X,F10.4,' TO',F10.4/' AVG=',F10.4,' STD DEV=',F10.4)
C
 RETURN
 END
C*** OF STAT
```

```
      SUBROUTINE NOISE (R,LR,LD,NA,AMP,Z)
C
C PUTS GAUSSIAN NOISE IN LINEAR ARRAY NA WITH AMP AS GIVEN
C (0 DEFAULTS TO 1), DERIVED FROM SEED Z (0...1). IF NA -VE
C NOISE IS ADDED TO EXISTING CONTENTS. IF Z -VE EXISTING SEED
C IS USED. SEE IMAGE GENERATION SYSTEM FOR ROUTINE RDG.
C
      REAL R(LR)
C
      N=IABS(NA)
      IF(JV(LR,LD,1.0,N,0.0,0).NE.1) RETURN
C
      A=AMP
      IF(A.EQ.0.0) A=1.0
      IF(Z.GE.0.0) CALL RDG(Z,1)
C
      DO 100 I=N*LD,N*LD+LD-1
        CALL RDG(RD,0)
        IF(NA.GE.0) THEN
          R(I)=A*RD
        ELSE
          R(I)=R(I)+A*RD
        ENDIF
  100 CONTINUE
C
      RETURN
      END
C*** OF NOISE
C
C
C
      SUBROUTINE MULT (R,LR,LD,NA,NB,NC,AMP)
C
C MULTIPIES LINEAR ARRAYS NA, NB TOGETHER POINT BY POINT
C AND ENTERS INTO ARRAY NC, SCALED BY AMP (0 DEFAULTS TO 1).
C
      REAL R(LR)
C
      IF(JV(LR,LD,3.0,NA,NB,NC,0).NE.1) RETURN
C
      A=AMP
      IF(A.EQ.0.0) A=1.0
      KA=NA*LD
      KB=NB*LD
      KC=NC*LD
C
      DO 100 I=0,LD-1
  100 R(KC+I)=A*R(KA+I)*R(KB+I)
C
      RETURN
      END
C*** OF MULT
```

```
      SUBROUTINE FNC (R,LR,LD,NAME,NA,PA,PB,PC,PD,PE,PF)
C
C PERFORMS MATHEMATICAL OPERATIONS POINT BY POINT TO
C GENERATE A LINEAR ARRAY, SPECIFIED BY A CHARACTER
C VALUE NAME AND USING 0,1, OR 2 INPUT LINEAR ARRAYS,
C AND UP TO 6 PARAMETERS.
C
C 0 INPUT ARRAY OPTIONS...SINX,COSX,DISC,POLY,QUINT
C 1 INPUT ARRAY OPTIONS...LOG,ALOG,LN,EXP,SQRT,SQ,RCP,MIN,
C                         MAX,CLIP,SIN,COS,DSZ,DS
C 2 INPUT ARRAY OPTIONS...MLT,DIV
C
      REAL R(LR)
      LOGICAL F
      F(K)=NAME.EQ.K
C
      N=1
      IF(F('SINX').OR.F('COSX').OR.F('DISC').OR.F('POLY').OR.
     +  F('QUINT')) N=0
      IF(F('DIV').OR.F('MLT')) N=2
C
      IF(JV(LR,LD,N+1.0,NA,INT(PA),INT(PB),0).NE.1) RETURN
C
      KA=NA*LD
      KB=INT(PA)*LD
      KC=INT(PB)*LD
C
      DO 100 I=0,LD-1
        X=FLOAT(I)/FLOAT(LD-1)
        A=0.0
C
C  ***NO INPUT ARRAY
C
        IF(F('SINX')) A=SIN(PA*X)
        IF(F('COSX')) A=COS(PA*X)
        IF(F('DISC')) THEN
          A=1.0
          IF(I.EQ.(LD-1)) A=0.5
        ENDIF
        IF(F('POLY')) THEN
          G=X**2
          A=((((PF*G+PE)*G+PD)*G+PC)*G+PB)*G+PA
        ENDIF
        IF(F('QUINT')) A=((((PF*X+PE)*X+PD)*X+PC)*X+PB)*X+PA
C
C  ***ONE INPUT ARRAY
C
        IF(F('LOG')) A=ALOG10(AMAX1(1.0E-99,R(KA+I)))
        IF(F('ALOG')) A=10.0**AMIN1(100.0,R(KA+I))
        IF(F('LN')) A=ALOG(AMAX1(1.0E-99,R(KA+I)))
        IF(F('EXP')) A=EXP(AMIN1(300.0,R(KA+I)))
        IF(F('SQRT')) A=SQRT(AMAX1(0.0,R(KA+I)))
        IF(F('SQ')) A=R(KA+I)**2
        IF(F('RCP')) A=1.0/SIGN(AMAX1(1.0E-99,ABS(R(KA+I))),R(KA+I))
        IF(F('MIN')) A=AMIN1(R(KA+I),PB)
        IF(F('MAX')) A=AMAX1(R(KA+I),PB)
        IF(F('CLIP')) A=AMAX1(AMIN1(PB,PC),AMIN1(AMAX1(PB,PC),R(KA+I)))
        IF(F('SIN')) A=SIN(R(KA+I))
        IF(F('COS')) A=COS(R(KA+I))
C  ***DOUBLE LINE, BRIGHT COMP AT 0, PARAMETERS SEP, WEAK,
C  ***BRIGHT (MIN 1-WEAK ), LOG SCALE (0 OFF), OFFSET
        IF(F('DSZ')) THEN
          C=FLOAT(I)-PF*FLOAT(LD-1)
          A=AMAX1((1.0-PC),PD)*RXT(R,LR,LD,NA,C)+
     +      PC*RXT(R,LR,LD,NA,(C-FLOAT(LD-1)*PB))
          IF(PE.NE.0.0) A=PE*ALOG10(AMAX1(1.0E-99,A))
        ENDIF
C  ***DOUBLE LINE, PARAMETERS SEPARATION,WEAK,BRIGHT
C  ***(MIN 1-WEAK),LOG SCALE
        IF(F('DS')) THEN
          W=PC
          B=AMAX1((1.0-PC),PD)
          A=W*RXT(R,LR,LD,NA,(FLOAT(I)-FLOAT(LD-1)*(0.5+PB*B)))+
     +      B*RXT(R,LR,LD,NA,(FLOAT(I)-FLOAT(LD-1)*(0.5-PB*W)))
          IF(PE.NE.0.0) A=PE*ALOG10(AMAX1(1.0E-99,A))
        ENDIF
C
C  ***TWO INPUT ARRAYS
C
        IF(F('MLT')) A=R(KA+I)*R(KB+I)
        IF(F('DIV')) THEN
          A=R(KA+I)
          B=R(KB+I)
          IF(B.EQ.0.0) B=1.0E-99
          A=A/B
        ENDIF
C
        IF(N.EQ.0) R(KA+I)=A
        IF(N.EQ.1) R(KB+I)=A
        IF(N.EQ.2) R(KC+I)=A
  100 CONTINUE
C
      RETURN
      END
C*** OF FNC
```

```
      SUBROUTINE TMIX (R,LR,LD,NTA,NTB,NTC,AA,AB,OFF)
C
C COPIES THE SUM OF TRIPLE LINEAR ARRAYS NTA,NTB INTO NTC,
C WEIGHTS AA,AB, PLUS AN OFFSET OFF.
C
      REAL R(LR)
C
      NH=MAXO(NTA,NTB,NTC)
      NL=MINO(NTA,NTB,NTC)
      IF(JV(LR,LD,2.0,NH+1,NL-1,0.0).NE.1) RETURN
C
      KA=(NTA-1)*LD
      KB=(NTB-1)*LD
      KC=(NTC-1)*LD
C
      DO 100 I=0,3*LD-1
  100 R(KC+I)=AA*R(KA+I)+AB*R(KB+I)+OFF
C
      RETURN
      END
C*** OF TMIX
C
C
C
      SUBROUTINE TNOISE (R,LR,LD,NT,AMP,Z)
C
C PUTS GAUSSIAN NOISE IN A TRIPLE LINEAR ARRAY NT WITH
C AMPLITUDE AMP (0 DEFAULTS TO 1.0) DERIVED FROM SEED Z (0...1).
C IF NA -VE NOISE IS ADDED TO EXISTING CONTENTS. IF Z -VE
C EXISTING SEED IS USED.
C
      REAL R(LR)
C
      N=IABS(NT)
      IF(JV(LR,LD,3.0,N-1,N,N+1.0).NE.1) RETURN
C
      INC=1
      IF(NT.LT.0) INC=-1
      CALL NOISE(R,LR,LD,NT-INC,AMP,Z)
      CALL NOISE(R,LR,LD,NT,AMP,-1.0)
      CALL NOISE(R,LR,LD,NT+INC,AMP,-1.0)
C
      RETURN
      END
C*** OF TNOISE
C
C
C
      SUBROUTINE TLINE (R,LR,LD,NP,NT,AA,XA,AB,XB,OFF)
C
C PUTS TWO LINES PROFILE NP INTO TRIPLE LINEAR ARRAY NT
C AMPLITUDE AND POSITION AA,XA  AB,XB  PLUS OFFSET OFF.
C POSITION IS MEASURED ALONG CENTRAL LINEAR ARRAY. IF NT
C -VE OUTPUT ARRAYS ARE NOT INITIALLY ZEROED. IF NP -VE
C A COMPLETE PROFILE IS ASSUMED RATHER THAN HALF, AND
C POSITION IS TO CENTRE OF NP.
C
      REAL R(LR)
C
      NPA=IABS(NP)
      NTA=IABS(NT)
      IF(JV(LR,LD,4.0,NPA,NTA-1,NTA,NTA+1).NE.1) RETURN
C
      KT=(NTA-1)*LD
      IF(NT.GT.0) THEN
        DO 100 I=0,LD*3-1
  100   R(KT+I)=OFF
      ELSE
        DO 110 I=0,LD*3-1
  110   R(KT+I)=R(KT+I)+OFF
      ENDIF
C
      A=AA
      X=XA
      DO 130 J=0,1
        IF(A.NE.0.0) THEN
          SC=FLOAT(LD)+X*FLOAT(LD-1)
          DO 120 I=0,LD*3-1
            S=FLOAT(I)-SC
            IF(NP.GT.0) THEN
              R(KT+I)=R(KT+I)+A*RXT(R,LR,LD,NPA,S)
            ELSE
              S=S+FLOAT(LD/2)
              IF(S.GE.0.0) R(KT+I)=R(KT+I)+A*RXT(R,LR,LD,NPA,S)
            ENDIF
  120     CONTINUE
        ENDIF
        A=AB
        X=XB
  130 CONTINUE
C
      RETURN
      END
C*** OF TLINE
      SUBROUTINE TCONV (R,LR,LD,NT,NF,NB,AMP,LF)
C
C CONVOLVES TRIPLE LINEAR ARRAY NT WITH LINEAR ARRAY NF TO
C FORM LINEAR ARRAY NB. AMP IS A SCALING FACTOR, 0 DEFAULTING
C TO 1. LF LIMITS THE WIDTH OF NF USED, FROM THE CENTRE OF THE
C PROFILE. IF NF -VE, THE CONVOLVING PROFILE IS CENTRED IN NF
C RATHER THAN BEING SYMMETRIC ABOUT THE ORIGIN.
C
      REAL R(LR)
C
      N=IABS(NF)
      IF(JV(LR,LD,4.0,NT-1,NT+1,N,NB).NE.1) RETURN
C
      KT=NT*LD
      KF=N*LD
      KB=NB*LD
      A=AMP
      IF(A.EQ.0.0) A=1.0
      L=MINO((LD-1),MAXO(0,LF))
      IF(L.EQ.0) L=LD-1
C
      DO 120 I=0,LD-1
        T=0.0
        IF(NF.GT.0) THEN
C         ***HALF PROFILE GIVEN
          DO 100 J=-L,L
  100     T=T+R(KT+I+J)*R(KF+IABS(J))
        ELSE
C         ***FULL PROFILE GIVEN
          L=MINO(LD/2,L)
          DO 110 J=LD/2-L,LD/2+L
  110     T=T+R(KT+I+(J-LD/2))*R(KF+J)
        ENDIF
        R(KB+I)=T*A
  120 CONTINUE
C
      RETURN
      END
C*** OF TCONV
```

```
      SUBROUTINE SQ (R,LR,LD,NA,NB)
C
C FORMS A SQUARE ARRAY NB FROM A LINEAR ARRAY NA BY
C USING IT AS THE X AND Y SEPARABLE COMPONENTS.
C
      REAL R(LR)
C
      IF(JV(LR,LD,1,1,NA,NB,0,0).NE.1) RETURN
C
      KA=NA*LD
      KB=NB*(LD**2)
C
      DO 110 IY=0,LD-1
        KY=KB+IY*LD
        DO 100 IX=0,LD-1
  100   R(KY+IX)=R(KA+IY)*R(KA+IX)
  110 CONTINUE
C
      RETURN
      END
C*** OF SQ
C
C
C
      SUBROUTINE RD (R,LR,LD,NA,NB,SCALE,AMP)
C
C FORMS A SQUARE ARRAY NB FROM A LINEAR ARRAY NA BY USING
C IT AS A RADIAL PROFILE, EXPANDING BY A FACTOR SCALE IN
C RADIUS, AND AMP IN AMPLITUDE. 0 DEFAULTING TO 1, AND
C SCALE -1 TO 1.414...
C
      REAL R(LR)
C
      IF(JV(LR,LD,1,1,NA,NB,0,0).NE.1) RETURN
C
      KB=NB*(LD**2)
      A=AMP
      IF(A.EQ.0.0) A=1.0
      S=SCALE
      IF(S.EQ.0.0) S=1.0
      IF(S.EQ.-1.0) S=SQRT(2.0)
C
      DO 110 IY=0,LD-1
        KY=KB+IY*LD
        DO 100 IX=0,LD-1
          X=SQRT(FLOAT(IX**2+IY**2))/S
          R(KY+IX)=A*RXT(R,LR,LD,NA,X)
  100   CONTINUE
  110 CONTINUE
C
      RETURN
      END
C*** OF RD
C
C
C
      SUBROUTINE LN (R,LR,LD,NA,NB)
C
C PROJECTS A SQUARE ARRAY NA ONTO A LINEAR ARRAY NB.
C
      REAL R(LR)
C
      IF(JV(LR,LD,1,1,NB,NA,0,0).NE.1) RETURN
C
      KA=NA*(LD**2)
      KB=NB*LD
      C=2.0/SQRT(FLOAT(2*LD-1))
C
      DO 110 IX=0,LD-1
        S=0.0
        KX=KA+IX
        DO 100 IY=0,LD-1
  100   S=S+R(KX+IY*LD)*C
        R(KB+IX)=S-0.5*R(KX)*C
  110 CONTINUE
C
      RETURN
      END
C*** OF LN
C
C
C
      SUBROUTINE COPYSQ (R,LR,LD,NA,NB,AMP,OFFSET)
C
C COPIES SQUARE ARRAY NA TO NB, MULTIPLYING BY AMP (0 DEFAULTS
C TO 1) AND ADDING OFFSET.
C
      REAL R(LR)
      IF(JV(LR,LD,0,2,NA,NB,0,0).NE.1) RETURN
C
      A=AMP
      IF(A.EQ.0.0) A=1.0
      KA=NA*(LD**2)
      KB=NB*(LD**2)
C
      DO 110 IY=0,LD-1
        KYA=KA+IY*LD
        KYB=KB+IY*LD
        DO 100 IX=0,LD-1
  100   R(KYB+IX)=A*R(KYA+IX)+OFFSET
  110 CONTINUE
C
      RETURN
      END
C*** OF COPYSQ
C
C
C
      SUBROUTINE MIXSQ (R,LR,LD,NA,NB,NC,AMPA,AMPB,OFFSET)
C
C COPIES THE SUM OF SQUARE ARRAYS NA, NB INTO NC WITH
C WEIGHTING FACTORS AMPA, AMPB (0 DEFAULTS TO 1) AND
C ADDING OFFSET.
C
      REAL R(LR)
C
      IF(JV(LR,LD,0,3,NA,NB,NC,0).NE.1) RETURN
C
      AA=AMPA
      IF(AA.EQ.0.0) AA=1.0
      AB=AMPB
      IF(AB.EQ.0.0) AB=1.0
      KA=NA*(LD**2)
      KB=NB*(LD**2)
      KC=NC*(LD**2)
C
      DO 110 IY=0,LD-1
        KYA=KA+IY*LD
        KYB=KB+IY*LD
        KYC=KC+IY*LD
        DO 100 IX=0,LD-1
  100   R(KYC+IX)=AA*R(KYA+IX)+AB*R(KYB+IX)+OFFSET
  110 CONTINUE
C
      RETURN
      END
C*** OF MIXSQ
C
C
C
      SUBROUTINE SCALSQ (R,LR,LD,NA,AMP,OFFSET)
C
C MULTIPLIES A SQUARE ARRAY IN PLACE BY AMP, ADDING OFFSET.
C
      REAL R(LR)
C
      IF(JV(LR,LD,0,1,NA,0,0,0).NE.1) RETURN
C
      KA=NA*(LD**2)
      DO 110 IY=0,LD-1
        KY=KA+IY*LD
        DO 100 IX=0,LD-1
  100   R(KY+IX)=R(KY+IX)*AMP+OFFSET
  110 CONTINUE
C
      RETURN
      END
C*** OF SCALSQ
C
C
C
      SUBROUTINE CONSQ (R,LR,LD,NA,NB,NC,AMP)
C
C FORMS A LINEAR ARRAY NC BY CONVOLVING SQUARE ARRAYS
C NA, NB. IF AMP IS NOT ZERO IT IS APPLIED AS A SCALING
C FACTOR.
C   USES 1 AS TEMP.
C
      REAL R(LR)
C
      IF(JV(LR,LD,1,2,NC,NA,NB,0).NE.1) RETURN
C
      CALL SCALE(R,LR,LD,NC,0.0,0.0,0.0,0.0)
      A=2.0/SQRT(FLOAT(2*LD-1))
      IF(AMP.NE.0.0) A=AMP
C
      DO 100 IY=0,LD-1
        NYA=NA*LD+IY
        NYB=NB*LD+IY
        CALL CONV(R,LR,LD,NYA,NYB,1,0.0)
        B=A
        IF(IY.EQ.0) B=0.5*A
        CALL MIX(R,LR,LD,1,NC,NC,B,1.0,0.0)
  100 CONTINUE
C
      RETURN
      END
C*** OF CONSQ
```

```
      SUBROUTINE FTRANS (R,LR,LD,NA,NB,SCALE)
C
C GENERATES THE DISCRETE COSINE TRANSFORM OF LINEAR ARRAY NA
C INTO NB. SCALE IF GREATER THAN 1 CAUSES THE PERIOD USED
C TO BE GREATER THAN 2*LD-1, DATA BEING ASSUMED 0 OUTSIDE
C RANGE GIVEN, AND THE HIGHER FREQUENCIES OF THE TRANSFORM
C NOT BEING CALCULATED.
C
      REAL R(LR)
C
      IF(JV(LR,LD,2,0,NA,NB,0,0).NE.1) RETURN
C
      M=2*LD-1
      IF(SCALE.GT.1.0) M=2*INT(SCALE*FLOAT(LD))-1
      C=SQRT(4.0/FLOAT(M))
      CF=2.0*3.1415926536/FLOAT(M)
      KA=NA*LD
      KB=NB*LD
C
      DO 110 IB=0,LD-1
        S=0.5*R(KA)
        DO 100 IA=1,LD-1
        100 S=S+R(KA+IA)*COS(CF*FLOAT(IA*IB))
        R(KB+IB)=C*S
      110 CONTINUE
C
      RETURN
      END
C*** OF FTRANS
C
C
C
      SUBROUTINE FTWO (R,LR,LD,NA,NB,SCAL)
C
C COSINE TRANSFORMS LINEAR ARRAY NA TO NB, INTERPRETED AS
C A TWO-D TRANSFORM OF A CIRCULAR PROFILE. SCAL GREATER
C THAN 1 EXPANDS IN OUTPUT SPACE.
C   1,SQ1 USED AS TEMP.
C
      REAL R(LR)
C
      IF(JV(LR,LD,2,0,NA,NB,0,0).NE.1) RETURN
C
      CALL RO(R,LR,LD,NA,1,0.0,0.0)
      CALL LN(R,LR,LD,1,1)
      CALL FTRANS(R,LR,LD,1,NB,SCAL)
      IF(SCAL.GT.1.0) THEN
        C=SQRT(FLOAT(2*LD-1)/FLOAT(2*INT(SCAL*FLOAT(LD))-1))
        CALL SCALE(R,LR,LD,NB,C,0.0,0.0)
      ENDIF
C
      RETURN
      END
C*** OF FTWO
C
C
C
      SUBROUTINE SPEC (R,LR,LD,IU,NA,NB,LL,NL,OFF)
C
C GENERATES A SPECTRUM IMAGE ON FILE IU, LINE LENGTH LL,
C NUMBER OF LINES NL, BY MULTIPLYING A SPECTRUM PROFILE IN
C LINEAR ARRAY NA BY A TRANSVERSE PROFILE IN NB, BOTH SCALED
C TO REQUIRED LENGTH, AND ADDING AN OFFSET VALUE OFF.
C   1,2,3 USED AS TEMP.
C
      REAL R(LR)
C
      J=1
      IF(IU.LE.0.OR.LL.LE.1.OR.NL.LE.1) J=0
      IF(JV(LR,LD,3,0,NA,NB,J,0).NE.1) RETURN
C
      CALL COPY(R,LR,LD,NA,1,0.0,0.0,FLOAT(LL-1)/FLOAT(LD-1))
      CALL COPY(R,LR,LD,NB,2,0.0,0.0,FLOAT(NL-1)/FLOAT(LD-1))
C
      DO 100 I=0,NL-1
        CALL MIX(R,LR,LD,1,1,3,R(2*LD+I),0.0,OFF)
        WRITE(IU) (R(J),J=3*LD,3*LD+LL-1)
      100 CONTINUE
C
      RETURN
      END
C*** OF SPEC
      SUBROUTINE FLTGEN (R,LR,LD,NA,LF,IOUT)
C
C WRITES A FILTER ARRAY OF LF BY LF POINTS TO UNIT
C IOUT, FORMED FROM SQUARE ARRAY NA. LF SHOULD BE ODD.
C   MAY USE 1 AS TEMP.
C
      REAL R(LR)
C
      IF(JV(LR,LD,2,1,LF,(2*LD-LF),NA,0).NE.1) RETURN
C
      KA=NA*(LD**2)
      KC=(LF+1)/2
C
      DO 110 ILN=1,LF
        IY=IABS(ILN-KC)
        KY=KA+IY*LD
C
        DO 100 IP=1,LF
          IX=IABS(IP-KC)
          R(IP)=R(KY+IX)
        100 CONTINUE
C
        WRITE(IOUT) (R(I),I=1,LF)
      110 CONTINUE
C
      RETURN
      END
C*** OF FLTGEN
```

```
      SUBROUTINE SAVE (R,LR,LD,IU,NA,NB)
C
C WRITES FROM BUFFER TO UNIT IU. IT STARTS WITH LINEAR ARRAY
C NA AND ENDS WITH NB. IF NB IS LESS THAN NA ONLY NA IS WRITEN.
C IF NA AND NB ARE BOTH ZERO THE ENTIRE USABLE BUFFER IS DUMPED.
C
      REAL R(LR)
C
      IF(IU.LE.0.OR.IU.GT.99) THEN
        WRITE(6,*) ' UNIT ERROR'
      ELSE
        NF=MAXO(NA,0)
        NL=MAXO(NF,NB)
        IF(NA.LE.0.AND.NB.LE.0) NL=(LR+1)/LD-1
        NMAX=(LR+1)/LD-1
        NF=MINO(NMAX,NF)
        NL=MINO(NMAX,NL)
        DO 100 I=NF,NL
          IF(I.EQ.0) THEN
            WRITE(IU) (R(IP),IP=1,LD)
          ELSEIF(I.LT.(LR+1)/LD) THEN
            WRITE(IU) (R(IP),IP=I*LD,(I*LD+LD-1))
          ENDIF
        100 CONTINUE
      ENDIF
C
      RETURN
      END
C*** OF SAVE
C
C
C
      SUBROUTINE REST (R,LR,LD,IU,NA,NB)
C
C READS INTO BUFFER FROM IU, INTO LINEAR ARRAYS FROM NA TO NB.
C IF NB IS LESS THAN NA, INTO NA ONLY. IF NA AND NB ARE BOTH
C ZERO, FROM THE START TO END OF DATA OR BUFFER.
C
      REAL R(LR)
C
      IF(IU.LE.0.OR.IU.GT.99) THEN
        WRITE(6,*) ' UNIT ERROR'
        RETURN
      ENDIF
C
      NF=MAXO(NA,0)
      NL=MAXO(NF,NB)
      IF(NA.LE.0.AND.NB.LE.0) NL=(LR+1)/LD-1
      NMAX=(LR+1)/LD-1
      NF=MINO(NMAX,NF)
      NL=MINO(NMAX,NL)
C
      DO 100 I=NF,NL
        IF(I.EQ.0) THEN
          READ(IU,END=110,ERR=120) (R(IP),IP=1,LD)
        ELSEIF(I.LT.(LR+1)/LD) THEN
          READ(IU,END=110,ERR=120) (R(IP),IP=I*LD,(I*LD+LD-1))
        ENDIF
      100 CONTINUE
      RETURN
C
      110 WRITE(6,*) ' END OF FILE'
      RETURN
C
      120 WRITE(6,*) ' READ ERROR'
      RETURN
      END
C*** OF REST
C
C
C
      FUNCTION JV (LR,LD,NLN,NSQ,NA,NB,NC,ND)
C
C RETURNS 1 IF ARRAYS NUMBERS NA...ND ARE VALID, THE
C NUMBERS THAT ARE TO BE LINEAR AND SQUARE BEING GIVEN
C AS NLN, NSQ, RETURNS 0 OTHERWISE AND WRITES A MESSAGE.
C
      INTEGER N(4)
C
      N(1)=NA
      N(2)=NB
      N(3)=NC
      N(4)=ND
C
      J=1
      IF(NLN.GE.1) THEN
        DO 100 I=1,NLN
        100 IF((N(I)+1)*LD.GT.LR.OR.N(I).LE.0) J=0
      ENDIF
      IF(NSQ.GE.1) THEN
        DO 110 I=NLN+1,NLN+NSQ
        110 IF((N(I)+1)*(LD**2).GT.LR.OR.N(I).LE.0) J=0
      ENDIF
      IF(J.EQ.0) WRITE(6,*) ' PARAMETER ERROR'
      JV=J
C
      RETURN
      END
C*** OF FUNCTION JV
      FUNCTION RXT (R,LR,LD,N,X)
C
C RETURNS A LINEAR INTERPOLATED VALUE FOR POSITION ABS(X)
C IN ARRAY N TAKEN AS LINEAR. X BEING IN GRID UNITS. IT
C RETURNS 0 IF OVERRANGE.
C
      REAL R(LR)
C
      AX=ABS(X)
      IX=INT(AX)
      IF(AX.GT.FLOAT(LD-1)) THEN
        RX=0.0
        IF(AX.LT.FLOAT(LD)) RX=(FLOAT(LD)-AX)*R((N+1)*LD-1)
      ELSEIF(AX.EQ.FLOAT(IX)) THEN
        RX=R(N*LD+IX)
      ELSE
        PX=AX-FLOAT(IX)
        RL=R(N*LD+IX)
        RH=R(N*LD+IX+1)
        RX=RL+PX*(RH-RL)
      ENDIF
      RXT=RX
C
      RETURN
      END
C*** OF FUNCTION RXT
C
C
C
      FUNCTION RBIG (R,LR,LD,N)
C
C RETURNS THE LARGEST ABSOLUTE VALUE IN LINEAR ARRAY N.
C
      REAL R(LR)
C
      IF(JV(LR,LD,1,0,N,0,0,0).NE.1) THEN
        RBIG=0.0
        RETURN
      ENDIF
C
      KA=N*LD
      B=0.0
      DO 100 I=0,LD-1
      100 B=AMAX1(B,ABS(R(KA+I)))
C
      RBIG=B
C
      RETURN
      END
C*** OF FUNCTION RBIG
C
C
C
      FUNCTION PSUM (R,LR,LD,NA,NB,IXP)
C
C SUMS LINEAR ARRAY NA OVER ITS ELEMENTS, WEIGHTED BY LINEAR
C ARRAY NB IF NB +VE, AND BY THE POSITION TO THE POWER IXP.
C IF IXP IS -VE THE POWER IS -IXP-1 AND THE LAST POINT IS
C WEIGHTED 0.5.
C
      REAL R(LR)
C
      IF(JV(LR,LD,2,0,NA,MAXO(1,NB),0,0).NE.1) THEN
        PSUM=0.0
        RETURN
      ENDIF
C
      N=MAXO(IXP,-(IXP+1))
      KA=NA*LD
      KB=NB*LD
      S=0.0
C
      DO 100 I=0,LD-1
C
        W=1.0
        IF(N.GT.0) W=FLOAT(MAXO(I,1))/FLOAT(LD-1)
        IF(N.GT.1) W=W**N
        IF(I.EQ.0) THEN
          IF(N.EQ.0) W=W*0.5
          IF(N.EQ.1) W=W*0.125
          IF(N.EQ.2) W=W*0.04
          IF(N.GE.3) W=0.0
        ENDIF
        IF(I.EQ.(LD-1).AND.IXP.LT.0) W=W*0.5
C
        A=R(KA+I)
        IF(NB.GT.0) A=A*R(KB+I)
        S=S+W*A
C
      100 CONTINUE
C
      PSUM=S
C
      RETURN
      END
C*** OF FUNCTION PSUM
```

A4 Programs: Command Interpreter and Generator

A4.1 Command Interpreter

The interpreter commands are listed with details of the parameters required for their basic actions (denoted k, l, m...). The routines are then listed by name only, and again with brief descriptive notes. The source code for the interpreter and its routines follows.

$from	:input from k, 0 restores to default.
$step	:as $from, but each line has to be confirmed, and can be altered.
$log	:causes input to be copied to k, turned off for 0.
$rec	:similar to $log, to a second, logging, file.
$term	:indicates terminal is available for messages.
$msg	:similar to $term.
$exe	:switches execution of commands off or on for k=0, 1
$time	:displays cp time, and if k=1, after all commands until turned off by k=0.
$end	:input reverts to default source; if already default, program stops.
$n	:sets 'oldname' to k (no other action).
*	:a comment line, the rest is ignored.

Layout commands: these are written to log file, or to file k if k greater than zero.

$gap	:writes a group of blank lines as gap in output.
$div	:writes a gap with a dividing line across it.
$page	:forces a new page.
$rule	:writes a line of characters l, length m (default to 64).
$char	:writes a single character l (without leading blank).
$text	:copies text after k, with leading blank.
$rewind	:rewinds file unit number k.
$eof	:writes eof to file k.
$back	:backspaces file k by l records (default 1).
$list	:rewinds file k and copies to file l or terminal.
$help	:displays help message for command k, or if k absent for the previous command. Just $ is equivalent.

```
(PROGRAM RUN)

SUBROUTINE CONLIN (K,L,VALUE,KALPHA,LT,NTOK,NALPHA)
SUBROUTINE CONTOK (K,L,M,N,VAL,KTOK)
SUBROUTINE LTOK   (K,L,N)
SUBROUTINE LGAP   (K,L,N)
FUNCTION LITT   (K)
FUNCTION LNCOPY (IUA,IUB)



(PROGRAM RUN)
  THIS IS THE COMMAND INTERPRETER PROGRAM. IT IS SHOWN WITH A
  MINIMAL INITIAL SECTION WHICH HAS TO BE EXPANDED TO SUIT A
  SPECIFIC APPLICATION.

SUBROUTINE CONLIN (K,L,VALUE,KALPHA,LT,NTOK,NALPHA)
  CONVERTS A LINE OF CHARACTERS K(L) TO A SERIES OF
  NUMERIC VALUES IN ARRAY VALUE(LT) IF POSSIBLE, OTHERWISE
  SETTING TO ZERO AND ENTERING THE SYMBOLIC TOKEN IN
  THE CORRESPONDING PLACE IN KALPHA(LT) INSTEAD OF BLANK.
  THE TOTAL NUMBER OF TOKENS IS RETURNED AS NTOK, AND THE
  NUMBER OF ENTRIES IN KALPHA THAT ARE NOT BLANK AS NALPHA.

SUBROUTINE CONTOK (K,L,M,N,VAL,KTOK)
  CONVERTS A SUBSTRING OF THE ARRAY OF CHARACTERS K(L),
  SELECTED BY POINTERS M,N TO FIRST CHARACTER AND CHARACTER
  AFTER LAST, TO A NUMERIC VALUE VAL IF POSSIBLE, SETTING
  ALPH TO BLANK, OR OTHERWISE RETURNING ZERO AND SETTING
  ALPH TO THE SUBSTRING, TRUNCATED TO 10 CHARACTERS OR
  BLANK FILLED AS NECESSARY. ALL VARIABLES STARTING WITH J
  ARE FLAGS, 0/1.

SUBROUTINE LTOK (K,L,N)
  LOCATES THE NEXT NON-SPACING CHARACTER IN A CHARACTER
  ARRAY K(L) AFTER POSITION N, SETTING N TO THE NEW
  POSITION. USED BY CONLIN TO FIND THE START OF THE NEXT
  TOKEN.

SUBROUTINE LGAP (K,L,N)
  FINDS THE NEXT SPACING CHARACTER IN ARRAY K(L) AFTER
  POSITION N, SETTING N TO THE NEW POSITION. USED BY CONLIN
  TO FIND THE END OF A TOKEN.

FUNCTION LITT (K)
  RETURNS ITS PARAMETER AS ITS VALUE.

FUNCTION LNCOPY (IUA,IUB)
  COPIES A LINE FROM UNIT IUA TO IUB IF POSSIBLE,
  AND RETURNS 1 IF SUCCESSFUL, OTHERWISE 0.
```

```
      PROGRAM RUN (INPUT,TAPE5=INPUT,OUTPUT,TAPE6=OUTPUT)
C
C COMMAND INTERPRETER PROGRAM WITH A MINIMAL INITIAL SECTION
C WHICH NEEDS EXPANDING TO SUIT A SPECIFIC APPLICATION.
C
      INTEGER I(40),IX(40),K(40),KX(40),LINE(136)
      REAL R(40),RX(40)
      COMMON/RUNVAR/I,K,R,LL,IT,KA,
     +  KBLANK,OLDTM,JEX,JINF,JHLP,JLST
      COMMON/ERR/JERR,IERR
      COMMON/SYSVAR/IUMSG,IULOG,IUSET,
     +  IUSTD,IURD,IUREC,ISYS(10)
      DATA JSTART,IUREF/1,5/
      EQUIVALENCE (I(1),IX(1)),(K(1),KX(1))
      EQUIVALENCE (R(1),RX(1)
C
C...POSITION FOR ANY INITIALISING CODE
C
      GO TO 9950
 9900 CONTINUE
C
      IF(LINE(IT).EQ.LITT('$').OR.LINE(IT).EQ.LITT('*')) THEN
        NTOK=0
C
C...POSITION FOR GENERATED CONDITIONALS (ONE GIVEN
C...AS AN EXAMPLE)
C
      ELSEIF(KA.EQ.LITT('NAME')) THEN
        IF(JEX.GT.0) CALL NAME(I(2),I(3),R(4),100,N,I(5))
C
C THIS IS THE CONSTANT PART OF THE COMMAND INTERPRETER
C
      ELSE
        JFAIL=1
      ENDIF
C
C***PROCESS AN INTERPRETER COMMAND
C
      IF(LINE(IT).EQ.LITT('$').AND.KA.NE.LITT('$NULL')) THEN
        IF(KA.EQ.LITT('$END')) THEN
          IF(IUSET.EQ.IUSTD) STOP
          IUSET=IUSTD
          IURD=IUSTD
        ELSEIF(KA.EQ.LITT('$STEP')) THEN
          IF(IUMSG.GT.0) JSTEP=1
          IUSET=I(2)
          IF(IUSET.LE.0) IUSET=IUSTD
          IURD=IUSET
        ELSEIF(KA.EQ.LITT('$FROM')) THEN
          JSTEP=0
          IUSET=I(2)
          IF(IUSET.LE.0) IUSET=IUSTD
          IURD=IUSET
        ELSEIF(KA.EQ.LITT('$LOG')) THEN
          IULOG=I(2)
          JLOG=I(3)
        ELSEIF(KA.EQ.LITT('$REC')) THEN
          IUREC=I(2)
        ELSEIF(KA.EQ.LITT('$TIME')) THEN
          JTIME=I(2)
        ELSEIF(KA.EQ.LITT('$MSG')) THEN
          IUMSG=I(2)
        ELSEIF(KA.EQ.LITT('$TERM')) THEN
          IUMSG=I(2)
          IF(IUMSG.EQ.0) IUMSG=6
          KLST=I(3)
          IF(KLST.LE.0) KLST=3
          KHLP=I(4)
          IF(KHLP.LE.0) KHLP=KLST
          KINF=I(5)
          IF(KINF.LE.0) KINF=KHLP
        ELSEIF(KA.EQ.LITT('$EXE')) THEN
          KEX=I(2)
        ELSEIF(KA.EQ.LITT('$REWIND')) THEN
          IUA=I(2)
          IF(JEX.GT.0) THEN
            IF(IUA.GT.0.AND.(IUA.NE.IUSET.OR.JSTEP.GT.0)) THEN
              REWIND IUA
            ELSEIF(IUMSG.GT.0) THEN
              WRITE(IUMSG,9902) IUA
 9902         FORMAT(' REWIND',I3,' IMPOSSIBLE')
            ENDIF
          ENDIF
        ELSEIF(KA.EQ.LITT('$BACK')) THEN
          IUA=I(2)
          IF(JEX.GT.0) THEN
            IF(IUA.GT.0.AND.(IUA.NE.IUSET.OR.JSTEP.GT.0)) THEN
              DO 9903 IUB=1,MAXO(1,I(3))
 9903         BACKSPACE IUA
            ENDIF
          ENDIF
        ELSEIF(KA.EQ.LITT('$EOF')) THEN
          IUA=I(2)
          IF(JEX.GT.0.AND.IUA.GT.0) ENDFILE IUA
        ELSEIF(KA.EQ.LITT('$LIST')) THEN
          IUA=I(2)
          IF(IUA.LT.0) IUA=-IUA
          IF(IUA.EQ.0) IUA=IULOG
          IUB=I(3)
          IF(IUB.LT.0) IUB=-IUB
          IF(IUB.EQ.0) IUB=IUMSG
          NCPYLN=I(4)
          IF(NCPYLN.LE.0.AND.IUB.EQ.IUMSG) NCPYLN=100
          IF(NCPYLN.LE.0) NCPYLN=1000
          IF(JEX.GT.0.AND.IUA.GT.0.AND.(IUA.NE.IUSET.OR.
     +       JSTEP.GT.0).AND.IUB.NE.IUSET) THEN
C           ***STANDARD COPY TO IUB
            REWIND IUA
            ICPYLN=0
            9904 ICPYLN=ICPYLN+1
            IF(LNCOPY(IUA,IUB).GT.0.AND.ICPYLN.LT.NCPYLN)
     +        GO TO 9904
            IF(ICPYLN.GE.NCPYLN.AND.IUMSG.GT.0)
     +        WRITE(IUMSG,9906) NCPYLN
 9906       FORMAT(' COPY REACHED',I4,' LINES LIMIT')
            IF(I(2).LT.0) REWIND IUA
            IF(I(3).LT.0) REWIND IUB
          ELSEIF(JEX.GT.0.AND.IUMSG.GT.0) THEN
C           ***REPORT FAILURE
            WRITE(IUMSG,9912) I(2),I(3)
 9912       FORMAT(' LIST',I3,' TO',I3,' IMPOSSIBLE')
          ENDIF
        ELSE
C         ***LAYOUT COMMANDS
          IUA=I(2)
          IF(IUA.LE.0) IUA=IULOG
          IF(KA.EQ.LITT('$PAGE')) THEN
            IF(IUA.GT.0) WRITE(IUA,9916)
 9916       FORMAT('1')
          ELSEIF(KA.EQ.LITT('$GAP')) THEN
            IF(IUA.GT.0) WRITE(IUA,9918)
 9918       FORMAT(1X///)
          ELSEIF(KA.EQ.LITT('$DIV')) THEN
            IF(IUA.GT.0) WRITE(IUA,9920)
 9920       FORMAT(1X///1X,64('*')///)
          ELSEIF(KA.EQ.LITT('$RULE')) THEN
            KRULE=K(3)
            IF(KRULE.EQ.KBLANK) KRULE=LITT('*')
            LRULE=I(4)
            IF(LRULE.LE.0.OR.LRULE.GT.135) LRULE=64
            DO 9922 IL=1,LRULE
 9922       LINE(IL)=KRULE
            IF(IUA.GT.0) WRITE(IUA,9924) (LINE(IL),IL=1,LRULE)
 9924       FORMAT(1X,135A1)
          ELSEIF(KA.EQ.LITT('$CHAR')) THEN
            IF(IUA.GT.0) WRITE(IUA,9926) K(3)
 9926       FORMAT(A1)
          ELSEIF(KA.EQ.LITT('$TEXT')) THEN
            CALL LGAP(LINE,136,IT)
            CALL LTOK(LINE,136,IT)
            CALL LGAP(LINE,136,IT)
            IF(IUA.GT.0) WRITE(IUA,9928) (LINE(IL),IL=
     +        MINO(135,IT+1),MINO(135,MAXO(LL,IT+1)))
 9928       FORMAT(1X,135A1)
          ELSE
            JFAIL=1
          ENDIF
        ENDIF
      ENDIF
C
C***OUTPUT CP TIME AND ITS INCREASE
C
      IF(KA.EQ.LITT('$TIME').OR.JTIME.GE.3.OR.(JTIME.GE.1.AND.
     +   IUSET.EQ.IUSTD).OR.JFAIL.GT.0) THEN
        TIME=SECOND(X)
        IF(IUMSG.GT.0) WRITE(IUMSG,9930) TIME,(TIME-OLDTM)
        IF(IULOG.GT.0.AND.(JTIME.EQ.2.OR.JTIME.EQ.4))
     +    WRITE(IULOG,9930) TIME,(TIME-OLDTM)
 9930   FORMAT(' CP TIME',F8.3,'    INC',F8.3)
        OLDTM=TIME
      ENDIF
C
C***REPORT FAILURE TO FIND COMMAND
C
      IF(JFAIL.GT.0) THEN
        IF(IUMSG.GT.0) WRITE(IUMSG,9940) KA
        IF(IULOG.GT.0) WRITE(IULOG,9940) KA
 9940   FORMAT(1X,A10,' NOT FOUND')
        IF(IUMSG.LE.0) STOP
        IUSET=IUSTD
        IURD=IUSTD
      ENDIF
C
C***REPORT ERROR FLAG SET
C
      IF(JERR.GT.0.AND.IUMSG.GT.0) WRITE(IUMSG,9945) JERR
 9945 FORMAT(' ***ERROR***  JERR =',I6)
C
C***START OF LOOP
C
 9950 CONTINUE
C
```

```
C***INITIALISE
C
 IF(JSTART.EQ.1) THEN
   JSTART=0
   JLOG=0
   JSTEP=0
   JTIME=0
   KEX=1
   KBLANK=LITT(' ')
   KOLD=LITT('$NULL')
   OLDTM=0.0
C
   KINF=0
   KHLP=0
   KLST=0
   KHINF=6
   KHHLP=6
   KHLST=6
C
   IUMSG=0
   IULOG=0
   IUSET=IUREF
   IUSTD=IUREF
   IURD=IUREF
   IUREC=0
 ENDIF
C
C***READ AND CONVERT LINE
C
 JEND=1
 READ(IURD,9952,END=9954) LINE
 9952 FORMAT (136A1)
 JEND=0
 9954 CONTINUE
C
 IF(JEND.GT.0) THEN
   LINE(1)=LITT('$')
   LINE(2)=LITT('E')
   LINE(3)=LITT('N')
   LINE(4)=LITT('D')
   DO 9956 IL=5,136
   9956 LINE(IL)=KBLANK
   IF(IUSET.NE.IUSTD) REWIND IUSET
 ENDIF
C
 IT=0
 CALL LTOK(LINE,136,IT)
 IT=MINO(IT,135)
 LL=137
 9958 LL=LL-1
 IF(LINE(LL).EQ.KBLANK.AND.LL.GT.1) GO TO 9958
 CALL CONLIN(LINE,136,R,K,40,NTOK,NALPHA)
 KA=K(1)
 IF(LINE(IT).EQ.LITT('*')) KA=LITT('$NULL')
 DO 9960 IL=1,40
 9960 I(IL)=IFIX(R(IL))
C
C***DISPLAY LINE
C
 IF(IUMSG.GT.0.AND.IURD.NE.IUSTD.AND.JSTEP.GT.0) THEN
   WRITE(IUMSG,9970) (LINE(IL),IL=1,LL)
   9970 FORMAT(136A1)
 ENDIF
C
C***RESTORE CONTROL PARAMETERS
C
 JINF=KINF
 JHLP=KHLP
 JLST=KLST
 JEX=KEX
 JFAIL=0
 IURD=IUSET
 IERR=IUMSG
 JERR=0
C
C***CONFIRM INPUT IF REQUIRED
C
 IF(JSTEP.GT.0) THEN
   WRITE(IUMSG,9972)
   9972 FORMAT(' CONFIRM? Y/N')
   READ(IUSTD,9971) KS
   9971 FORMAT(A1)
   IF(KS.NE.LITT('Y')) THEN
C    ***NOT CONFIRMED, DEFAULT ACTION SKIP
     KA=LITT('$NULL')
     LINE(IT)=LITT('$')
     IF(KS.EQ.LITT('R')) THEN
C      ***REPLACE WITH LINE FROM TERMINAL
       IURD=IUSTD
     ELSEIF(KS.NE.LITT('S')) THEN
C      ***INPUT REVERTS TO TERMINAL
       IURD=IUSTD
       IUSET=IUSTD
     ENDIF
   ENDIF
 ENDIF
C
C***HELP FUNCTION
C
 IF(KA.EQ.LITT('$HELP').OR.KA.EQ.LITT('$')) THEN
   KA=K(2)
   JEX=0
   IF(I(2).GT.0) THEN
     KA=LITT('$NULL')
C    ***RESET 'HELP' PARAMETERS
     KHLST=I(2)
     IF(I(3).GT.0) KHLST=I(3)
     KHHLP=I(2)
     IF(I(4).GT.0) KHHLP=I(4)
     KHINF=I(2)
     IF(I(5).GT.0) KHINF=I(5)
   ELSE
     LINE(IT)=KBLANK
     JLST=KHLST
     IF(I(3).GT.0) JLST=I(3)
     JHLP=KHHLP
     IF(I(4).GT.0) JHLP=I(4)
     JINF=KHINF
     IF(I(5).GT.0) JINF=I(5)
   ENDIF
 ENDIF
C
C***NAME FUNCTION
C
 IF(KA.EQ.LITT('$N')) THEN
   KA=K(2)
   LINE(IT)=KBLANK
   JEX=0
   JHLP=0
   JINF=0
   JLST=0
 ENDIF
C
C***RESTORE AND SAVE COMMAND
C
 IF(KA.EQ.KBLANK) THEN
   KA=KOLD
   DO 9974 IL=1,39
     I(41-IL)=I(40-IL)
     K(41-IL)=K(40-IL)
     R(41-IL)=R(40-IL)
   9974 CONTINUE
   I(1)=0
   K(1)=KBLANK
   R(1)=0.0
 ENDIF
 IF(LINE(IT).NE.LITT('$').AND.LINE(IT).NE.LITT('*')) THEN
   KOLD=KA
 ENDIF
C
C***KILL OFF ANY MESSAGE IF NO MESSAGE UNIT
C
 IF(IUMSG.LE.0) THEN
   JHLP=0
   JINF=0
   JLST=0
 ENDIF
C
C***OUTPUT TO LOG FILES
C
 IF(IUREC.GT.0.AND.KA.NE.LITT('$NULL'))
+  WRITE(IUREC,9982) (LINE(IL),IL=1,LL)
 IF(IULOG.GT.0.AND.KA.NE.LITT('$NULL')) THEN
C  ***RESTORE $ IF DELETED
   KS=LINE(IT)
   IF(LINE(IT).EQ.KBLANK) LINE(IT)=LITT('$')
   IF(LINE(IT).EQ.LITT('*')) THEN
C    ***COMMENT LINE
     IF(JLOG.GE.3) THEN
       WRITE(IULOG,9980) (LINE(IL),IL=1,LL)
     ELSEIF((LL-IT).EQ.1.AND.LINE(LL).NE.LITT('$')) THEN
       WRITE(IULOG,9980) LINE(LL)
     ELSE
       LINE(IT)=KBLANK
       WRITE(IULOG,9980) (LINE(IL),IL=1,LL)
       LINE(IT)=LITT('*')
     ENDIF
     9980 FORMAT(136A1)
   ELSEIF(JLOG.GE.3) THEN
C    ***EXACT COPY
     WRITE(IULOG,9980) (LINE(IL),IL=1,LL)
   ELSEIF(JLOG.EQ.2.OR.LINE(IT).NE.LITT('$').OR.(JLOG.EQ.1.AND.
+      (KA.EQ.LITT('$STEP').OR.KA.EQ.LITT('$FROM')))) THEN
     WRITE(IULOG,9982) (LINE(IL),IL=1,MINO(135,LL))
     9982 FORMAT(1X,135A1)
   ENDIF
   LINE(IT)=KS
 ENDIF
C
C***JUMP TO GENERATED CONDITIONALS SECTION
C
 GO TO 9900
C
 END
C*** OF COMMAND INTERPRETER PROGRAM
```

```
      SUBROUTINE CONLIN (K,L,VALUE,KALPHA,LT,NTOK,NALPHA)
C
C CONVERTS A LINE OF CHARACTERS K(L) TO A SERIES OF
C NUMERIC VALUES IN ARRAY VALUE(LT) IF POSSIBLE, OTHERWISE
C SETTING TO ZERO AND ENTERING THE SYMBOLIC TOKEN IN
C THE CORRESPONDING PLACE IN KALPHA(LT) INSTEAD OF BLANK.
C THE TOTAL NUMBER OF TOKENS IS RETURNED AS NTOK, AND THE
C NUMBER OF ENTRIES IN KALPHA THAT ARE NOT BLANK AS NALPHA.
C
      REAL VALUE(LT)
      INTEGER K(L),KALPHA(LT)
      DATA KBLANK/1H /
C
      NTOK=0
      NALPHA=0
      I=0
      DO 100 IT=1,LT
        VALUE(IT)=0.0
        KALPHA(IT)=KBLANK
  100 CONTINUE
C
C***LOOP OVER TOKENS TILL LINE PROCESSED OR OUTPUT FULL
C
  110 CONTINUE
C  ***SET I TO START OF NEXT TOKEN
      CALL LTOK(K,L,I)
C  ***RETURN IF NO TOKEN FOUND
      IF(I.GT.L) RETURN
C  ***COUNT TOKEN AND SAVE START POSITION
      NTOK=NTOK+1
      IS=I
C  ***SET I TO START OF FOLLOWING GAP
      CALL LGAP(K,L,I)
C  ***CONVERT TOKEN
      CALL CONTOK(K,L,IS,I,VNUM,KTOK)
C  ***COUNT IF AN ALPHA TOKEN
      IF(KTOK.NE.KBLANK) NALPHA=NALPHA+1
C  ***COPY INTO OUTPUT ARRAYS AND RETURN IF FULL
      VALUE(NTOK)=VNUM
      KALPHA(NTOK)=KTOK
      IF(NTOK.GE.LT) RETURN
  GO TO 110
C
      END
C*** OF CONLIN
C
C
C
      SUBROUTINE CONTOK (K,L,M,N,VAL,KTOK)
C
C CONVERTS A SUBSTRING OF THE ARRAY OF CHARACTERS K(L),
C SELECTED BY POINTERS M,N TO FIRST CHARACTER AND CHARACTER
C AFTER LAST, TO A NUMERIC VALUE VAL IF POSSIBLE, SETTING
C ALPH TO BLANK, OR OTHERWISE RETURNING ZERO AND SETTING
C ALPH TO THE SUBSTRING, TRUNCATED TO 10 CHARACTERS OR
C BLANK FILLED AS NECESSARY. ALL VARIABLES STARTING WITH J
C ARE FLAGS, 0/1.
C
      INTEGER K(L)
      DATA KBLANK/1H /
C
      JNUM=1
      JNEG=0
      JPOINT=0
      ND=0
      NP=0
      NUM=0
      JEXP=0
      JNEGXP=0
      NDEXP=0
      NUMEXP=0
C
C***LOOP OVER CHARACTERS WHILE NUMBER FLAG STILL SET
C
      DO 100 I=M,(N-1)
      IF(JNUM.GT.0) THEN
C  ***CONVERT CHAR TO DIGIT VALUE, OTHERWISE NEGATIVE
        ID=SHIFT(K(I),-54).AND.77B
        ID=ID-27
        IF(ID.GT.9) ID=ID-64
C
        IF(JEXP.LE.0) THEN
C    ***NOT IN EXPONENT
          IF(ID.GE.0) THEN
C       ***DIGIT
            ND=ND+1
            NP=NP+1
            NUM=10*NUM+ID
          ELSEIF(I.EQ.M.AND.ID.EQ.-53) THEN
C       ***LEADING -
            JNEG=1
          ELSEIF(I.EQ.M.AND.ID.EQ.-54) THEN
C       ***LEADING +
            JNEG=0
          ELSEIF(JPOINT.LE.0.AND.ID.EQ.-44) THEN
C       ***FIRST DECIMAL POINT
            JPOINT=1
            NP=0
          ELSEIF(ND.GT.0.AND.ID.EQ.-22) THEN
C       ***E WITH AT LEAST ONE PRECEDING DIGIT
            JEXP=1
            IEXP=I
          ELSE
C       ***CHAR ILLEGAL IN NUMBER
            JNUM=0
          ENDIF
        ELSE
C    ***IN EXPONENT
          IF(ID.GE.0) THEN
C       ***DIGIT
            NDEXP=NDEXP+1
            NUMEXP=10*NUMEXP+ID
          ELSEIF(I.EQ.(IEXP+1).AND.ID.EQ.-53) THEN
C       ***- SIGN FOLLOWING E
            JNEGXP=1
          ELSEIF(I.EQ.(IEXP+1).AND.ID.EQ.-54) THEN
C       ***+ SIGN FOLLOWING E
            JNEGXP=0

          ELSE
C       ***CHARACTER ILLEGAL IN EXPONENT
            JNUM=0
          ENDIF
        ENDIF
      ENDIF
  100 CONTINUE
C
C***CHECK NUMBER AND ANY EXPONENT HAVE DIGITS
C
      IF(ND.LE.0.OR.(JEXP.GT.0.AND.NDEXP.LE.0)) JNUM=0
C
C***FORM OUTPUT VALUES
C
      IF(JNUM.GT.0) THEN
C  ***NUMBER TOKEN
        VAL=FLOAT(NUM)
C  ***SCALE IF A DECIMAL POINT
        IF(JPOINT.GT.0.AND.NP.GT.0) THEN
          DO 110 I=1,NP
            VAL=VAL*0.1
  110     CONTINUE
        ENDIF
C  ***CHANGE SIGN IF NEGATIVE
        IF(JNEG.GT.0) VAL=-VAL
C  ***SCALE IF A NON ZERO EXPONENT
        IF(JEXP.GT.0.AND.NUMEXP.GT.0) THEN
          DO 120 I=1,NUMEXP
            IF(JNEGXP.GT.0) VAL=VAL*0.1
            IF(JNEGXP.LE.0) VAL=VAL*10.0
  120     CONTINUE
        ENDIF
C  ***SET CHARACTER OUTPUT TO BLANK
        KTOK=KBLANK
      ELSE
C  ***ALPHA TOKEN
        DO 130 I=1,10
          KCH=KBLANK
          IF((M+I-1).LT.N) KCH=K(M+I-1)
C    ***FORM COMBINING MASK
          MK=SHIFT(77B,6*(10-I))
C    ***INSERT KCH IN OUTPUT PARAMETER
          KTOK=(SHIFT(KCH,-6*(I-1)).AND.MK).OR.(KTOK.AND..NOT.MK)
  130CONTINUE
C  ***SET NUMERIC OUTPUT TO ZERO
        VAL=0.0
      ENDIF
C
      RETURN
      END
C*** OF CONTOK
```

```
 SUBROUTINE LTOK (K,L,N)
C
C LOCATES THE NEXT NON-SPACING CHARACTER IN A CHARACTER
C ARRAY K(L) AFTER POSITION N, SETTING N TO THE NEW
C POSITION. USED BY COMLIN TO FIND THE START OF THE NEXT
C TOKEN.
C
 INTEGER K(L)
 DATA KBLANK,KCOMMA/1H ,1H,/
C
 100 N=N+1
   IF(N.GT.L) RETURN
   IF(K(N).NE.KBLANK.AND.K(N).NE.KCOMMA) RETURN
 GO TO 100
C
 END
C*** OF LTOK
C
C
C
 SUBROUTINE LGAP (K,L,N)
C
C FINDS THE NEXT SPACING CHARACTER IN ARRAY K(L) AFTER
C POSITION N, SETTING N TO THE NEW POSITION. USED BY COMLIN
C TO FIND THE END OF A TOKEN.
C
 INTEGER K(L)
 DATA KBLANK,KCOMMA/1H ,1H,/
C
 100 N=N+1
   IF(N.GT.L) RETURN
   IF(K(N).EQ.KBLANK.OR.K(N).EQ.KCOMMA) RETURN
 GO TO 100
C
 END
C*** OF LGAP
C
C
C
 FUNCTION LITT (K)
C
C RETURNS ITS PARAMETER AS ITS VALUE.
C
 LITT=K
 RETURN
 END
C*** OF LITT
C
C
C
 FUNCTION LNCOPY (IUA,IUB)
C
C COPIES A LINE FROM UNIT IUA TO IUB IF POSSIBLE,
C AND RETURNS 1 IF SUCCESSFUL, OTHERWISE 0.
C
 INTEGER LINE(136)
 DATA KBLANK/1H /
C
 JCPY=0
 IF(IUA.GT.0) THEN
   READ(IUA,1100,END=120) LINE
   1100 FORMAT(136A1)
   LL=137
   110 LL=LL-1
   IF(LINE(LL).EQ.KBLANK.AND.LL.GT.1) GO TO 110
   IF(LL.EQ.1.AND.LINE(1).EQ.KBLANK) LL=0
   IF(IUB.GT.0.AND.LL.GT.0) WRITE(IUB,1100) (LINE(IL),IL=1,LL)
   IF(IUB.GT.0.AND.LL.EQ.0) WRITE(IUB,1100)
   JCPY=1
 ENDIF
 LNCOPY=JCPY
 RETURN
C
 120 CONTINUE
 BACKSPACE IUA
 LNCOPY=JCPY
 RETURN
C
 END
C*** OF LNCOPY
```

A4.2 Generator Program

Routines are listed by name only, and then with brief descriptive notes. The source code follows.

```
PROGRAM GEN (TAPE5,TAPE6,TAPE7,TAPE8)

SUBROUTINE ADDPRG (IG)
SUBROUTINE ADDSR  (KA,IG)
SUBROUTINE ADDSEG (IT,IG)
SUBROUTINE ADDHLP (IT)
SUBROUTINE ADDINF (IT)
SUBROUTINE ADDDEC
SUBROUTINE ADDCON
SUBROUTINE ADDEND (JSFL)
SUBROUTINE COPYLN (NS,LL)

SUBROUTINE OPENPC (JDEF)
SUBROUTINE COMPPC (JCMPL)
SUBROUTINE ENDPC
SUBROUTINE FLSPC  (K,IG,JDEF)

SUBROUTINE OPENPR
SUBROUTINE LIBPR  (IG)
SUBROUTINE ENDPR
SUBROUTINE CMPSET (IG,JCMPL)

SUBROUTINE CMDWRT (IU,KP,NP)
SUBROUTINE CMDAB  (IU,KA,KB)
SUBROUTINE CMDABC (IU,KA,KB,KC)

SUBROUTINE TXTLNA (N,KLINE,LKL)
SUBROUTINE TXTLNB (N)
SUBROUTINE TXTLNE (N,KCH)
SUBROUTINE TXTLNL (N)
SUBROUTINE TXTLNP (IU,JP)
SUBROUTINE TXTLNW (N,NL,KCH)
SUBROUTINE WRDEXP (KWRD,KCHS,LCHS)

SUBROUTINE CNTDIG (K,LK,N,NL,ND)
SUBROUTINE LTOK   (K,L,N)
SUBROUTINE LGAP   (K,L,N)
FUNCTION LIT    (KWRD)
FUNCTION NDIGIT (KCH)
LOGICAL FUNCTION Q (KA,KB)
```

```
PROGRAM GEN (TAPE5,TAPE6,TAPE7,TAPE8)
  THIS PROGRAM IS DESIGNED TO ASSEMBLE THE FILES NEEDED TO
  PERFORM COMPLETE TASKS. THE RESULTING PROGRAM CAN BE DRIVEN
  BY A TEXT FILE OF COMMANDS, OR INTERACTIVELY FROM A TERMINAL.
  FILES REQUIRED ARE, IN ORDER :
    INPUT (CAN BE FROM TERMINAL)
    OUTPUT PARTIAL COMMAND INTERPRETER SOURCE FILE
    OUTPUT PROC FILE TO COMPLETE AND COMPILE SOURCE FILE
    OUTPUT PROC FILE TO RUN THE COMPILED FILE

SUBROUTINE ADDPRG (IG)
  GENERATES A PROGRAM STATEMENT AS SPECIFIED BY A '$P' LINE.

SUBROUTINE ADDSR (KA,IG)
  GENERATES A CONDITIONAL LINE CALLING A SUBROUTINE, AND A
  CONDITIONAL WRITE LINE TO DISPLAY ITS PARAMETER SPECIFICATION
  MESSAGE.

SUBROUTINE ADDSEG (IT,IG)
  GENERATES THE CONDITIONAL LINE STARTING A CONDITIONAL
  PROGRAM SEGMENT.

SUBROUTINE ADDHLP (IT)
  GENERATES A CONDITIONAL STATEMENT TO WRITE A 'HELP' MESSAGE.

SUBROUTINE ADDINF (IT)
  GENERATES A CONDITIONAL STATEMENT TO WRITE AN 'INFORMATION'
  MESSAGE.

SUBROUTINE ADDDEC
  WRITES THE STANDARD DECLARATIONS TO THE GENERATED SOURCE
  FILE.

SUBROUTINE ADDCON
  WRITES THE STANDARD STATEMENTS AT THE START OF THE
  CONDITIONALS SECTION OF THE SOURCE FILE.

SUBROUTINE ADDEND (JSFL)
  ENDS THE GENERATED SOURCE FILE.

SUBROUTINE COPYLN (NS,LL)
  OUTPUTS LINE FROM CHARACTER AFTER NS TO UNIT 6, UP TO LINE
  LENGTH LIMIT AND SUPRESSING TRAILING BLANKS, AND RETURNS
  POINTER TO LAST CHARACTER OF INPUT LINE AS LL.

SUBROUTINE OPENPC (JDEF)
  WRITES THE OPENING COMMANDS TO FILE PC.

SUBROUTINE COMPPC (JCMPL)
  WRITES THE COMPILER COMMAND TO FILE PC.

SUBROUTINE ENDPC
  WRITES THE CLOSING COMMANDS TO FILE PC.

SUBROUTINE FLSPC (K,IG,JDEF)
  WRITES THE COMMANDS TO FILE PC TO ACCESS AND APPEND THE
  FILES CALLED FOR IN A 'FILE' LINE.
```

```
SUBROUTINE OPENPR
  OPENS THE PR FILE. DUMMY.

SUBROUTINE LIBPR (IG)
  GENERATES STATEMENTS TO FILE PR TO CAUSE USE OF SPECIFIED
  LIBRARY. DUMMY.

SUBROUTINE ENDPR
  ENDS FILE PR. DUMMY.

SUBROUTINE CMPSET (IG,JCMPL)
  SETS PARAMETERS CONTROLLING COMPILER TO BE USED FROM
  INPUT LINE.

SUBROUTINE CMDWRT (IU,KP,NP)
  WRITES THE CONTENTS OF KP(NP) TO UNIT IU AS A COMMAND
  STATEMENT WITH ',' SEPARATORS AND NO EMBEDDED BLANKS.
  IF NP=1 A TRAILING '.' IS ADDED. IF AN ITEM ENDS IN
  '=' THE FOLLOWING ',' IS SUPRESSED. IF IT IS '=' THE
  PRECEDING ',' IS ALSO SUPRESSED.

SUBROUTINE CMDAB (IU,KA,KB)
  WRITES A COMMAND STATEMENT TO IU FORMED FROM TWO WORDS KA,KB.

SUBROUTINE CMDABC (IU,KA,KB,KC)
  WRITES A COMMAND STATEMENT TO IU FORMED OF THREE WORDS.

SUBROUTINE TXTLNA (N,KLINE,LKL)
  'ADD'. COPIES ANY NON BLANK CHARACTERS IN KLINE(LKL) INTO
  LINE STARTING WITH POSITION N+1.

SUBROUTINE TXTLNB (N)
  'BLANK'. SETS LINE TO BLANK AFTER THE FIRST N CHARACTERS.

SUBROUTINE TXTLNE (N,KCH)
  'ENTER'. COPIES ANY NON BLANK CHARACTERS IN A WORD KCH INTO
   LINE STARTING AT POSITION N+1.

SUBROUTINE TXTLNL (N)
  'LENGTH'. RETURNS THE POSITION OF THE LAST NON BLANK
   CHARACTER IN LINE.

SUBROUTINE TXTLNP (IU,JP)
  'PUT'. WRITES LINE TO UNIT IU. IF JP IS NON ZERO, A
   CHARACTER IS PREFIXED TO THE LINE. 1=BLANK 2=ONE 3=PLUS

SUBROUTINE TXTLNW (N,NL,KCH)
  'WORD'. TAKES CHARACTERS IN LINE FROM N+1 TO NL-1 OR TO A
  MAXIMUM OF 10 AND PACKS THEM INTO KCH WITH BLANK FILL.

SUBROUTINE WRDEXP (KWRD,KCHS,LCHS)
  EXPANDS A CDC WORD KWRD INTO SEPARATE CHARACTERS IN KCHS(LCHS).
```

```
SUBROUTINE CNTDIG (K,LK,N,NL,ND)
  RETURNS THE NUMBER OF CHARACTERS ND THAT ARE DIGITS BETWEEN
  N AND NL EXCLUSIVE (NL.GT.N) IN K(LK).

SUBROUTINE LTOK (K,L,N)
  LOCATES THE NEXT NON-SPACING CHARACTER IN A CHARACTER
  ARRAY K(L) AFTER POSITION N, SETTING N TO THE NEW
  POSITION. USED BY CONLIN TO FIND THE START OF THE NEXT
  TOKEN.

SUBROUTINE LGAP (K,L,N)
  FINDS THE NEXT SPACING CHARACTER IN ARRAY K(L) AFTER
  POSITION N, SETTING N TO THE NEW POSITION. USED BY CONLIN
  TO FIND THE END OF A TOKEN.

FUNCTION LIT (KWRD)
  RETURNS THE VALUE OF ITS PARAMETER. TO PERMIT GENERAL USE
  OF CHARACTER LITERALS.

FUNCTION NDIGIT (KCH)
  RETURNS THE INTEGER VALUE OF FIRST DIGIT OF KCH. IF CHARACTER
  NOT A DIGIT, RETURNS A VALUE .LT.0 UNIQUE TO CHARACTER.

LOGICAL FUNCTION Q (KA,KB)
  RETURNS TRUE IF KA.EQ.KB
```

```
      PROGRAM GEN (TAPE5,TAPE6,TAPE7,TAPE8)
C
C THIS PROGRAM IS DESIGNED TO ASSEMBLE THE FILES NEEDED TO
C PERFORM COMPLETE TASKS. THE RESULTING PROGRAM CAN BE DRIVEN
C BY A TEXT FILE OF COMMANDS, OR INTERACTIVELY FROM A TERMINAL.
C
C FILES REQUIRED ARE, IN ORDER -
C   INPUT (CAN BE FROM TERMINAL)
C   OUTPUT PARTIAL COMMAND INTERPRETER SOURCE FILE
C   OUTPUT PROC FILE TO COMPLETE AND COMPILE SOURCE FILE
C   OUTPUT PROC FILE TO RUN THE COMPILED FILE
C
C ASSEMBLES THREE FILES, A SOURCE FILE, A PROCEDURE FILE TO
C CONSTRUCT A COMPLETE OBJECT FILE FROM THE SOURCE FILE AND THE
C SPECIFIED FILES OF SUBROUTINES, AND A PROCEDURE FILE TO RUN
C THE OBJECT FILE USING SPECIFIED LIBRARIES. THE INPUT IS A FILE
C OF TEXT LINES WHICH MAY BE LINE NUMBERED, AND CAN INCLUDE
C FORTRAN STATEMENTS FOR INCLUSION IN THE SOURCE FILE AS WELL
C AS LINES INTERPRETED BY THIS PROGRAM.
C
C THESE ARE INDEX VARIABLES USED IN GEN:
C   NS :POINTS BEFORE FIRST CHARACTER AFTER REMOVING LINE NO
C   IS :ADVANCED BY 2 FROM NS IF A LEADING 'C '
C   IT :POINTS TO START OF FIRST TOKEN STARTING AFTER IS
C   IG :POINTS TO START OF FOLLOWING GAP
C   LNO:THE NEXT LINE NUMBER TO BE USED, IN /LINENO/
C
      DATA JSFL,JPCFL,JPRFL,JCT,JCMPL,JDEF/0,0,0,0,0,1/
      COMMON/TXTLN/LINE(136)
      COMMON/LINENO/LNO
      LOGICAL Q
C
      LNO=10000
      REWIND 6
      REWIND 7
      REWIND 8
C
  100 CONTINUE
C
C  ***INPUT VALID LINE, SKIPPING REMARKS   /1/
C
    110 CONTINUE
        JEND=1
        READ(5,1,END=120) LINE
      1 FORMAT(136A1)
        JEND=0
  120   CONTINUE
C       ***STRIP OFF ANY 5 DIGIT LINE NUMBER AND
C       ***SET POINTER NS BEFORE FIRST WANTED CHARACTER
        CALL CNTDIG(LINE,136,0,5,ND)
        NS=0
        IF(ND.EQ.5) NS=5
C       ***GET NEXT LINE IF A REMARK AND NOT AT END OF FILE
    IF(Q(LINE(NS+1),'*').AND.JEND.LE.0) GO TO 110
C
C
C  ***SET IS TO BEFORE FIRST CHARACTER AFTER STRIPPING   /2/
C  ***ANY LEADING 'C ', IT TO START OF FIRST TOKEN, IG TO
C  ***START OF FOLLOWING GAP, AND ISOLATE FIRST TOKEN K
C
C
    IS=NS
    IF(Q(LINE(NS+1),'C').AND.Q(LINE(NS+2),' ')) IS=NS+2
    IG=IS
    CALL LTOK(LINE,136,IG)
    IT=IG
    CALL LGAP(LINE,136,IG)
    CALL TXTLNW(IT-1,IG,K)
C
C
C  ***SET LINE TYPE LT   /3/
C
C
    IF(Q(K,'$END').OR.JEND.GT.0) THEN
      LT=LIT('END')
    ELSEIF(Q(K,'$NODEF')) THEN
      LT=LIT('NODEF')
    ELSEIF(Q(K,'$L')) THEN
      LT=LIT('LIB')
    ELSEIF(Q(K,'$C')) THEN
      LT=LIT('COMPILER')
    ELSEIF(Q(K,'$S').OR.Q(K,'$SL').OR.
   +       Q(K,'$B').OR.Q(K,'$BL')) THEN
      LT=LIT('FILE')
    ELSEIF(Q(LINE(NS+1),'C').AND.JSFL.NE.4) THEN
      LT=LIT('COMMENT')
    ELSEIF(Q(K,' ')) THEN
      LT=LIT('NULL')
    ELSE
      LT=LIT('PROG')
    ENDIF
C
C
C  ***SET PARAMETERS FROM A 'CONTROL' LINE   /4/
C
C
    IF(Q(LT,'COMPILER')) THEN
      CALL CMPSET(IG,JCMPL)
    ENDIF
    IF(Q(LT,'NODEF')) THEN
      JDEF=0
    ENDIF
C
C
C  ***PROCESS GENERATED SOURCE FILE   /5/
C
C
C  ***COPY A COMMENT LINE DIRECTLY
    IF(Q(LT,'COMMENT')) THEN
      CALL COPYLN(NS,LL)
C
C  ***CLOSE FILE ON 'END' ACCORDING TO CURRRENT STATUS
    ELSEIF(Q(LT,'END')) THEN
      CALL ADDEND(JSFL)
C
C  ***OR PROCESS A PROGRAM LINE
    ELSEIF(Q(LT,'PROG')) THEN
C
      IF(JSFL.EQ.0) THEN
C       ***SOURCE FILE STATUS IS 'EMPTY'
C       ***LINE IS FIRST OF TYPE 'PROGRAM'
        IF(Q(K,'$P')) THEN
C         ***PROGRAM SPECIFICATION LINE
          CALL ADDPRG(IG)
          LT=LIT('NULL')
          JSFL=2
        ELSEIF(Q(K,'PROGRAM')) THEN
C         ***QUOTED (STANDARD) PROGRAM LINE
          CALL COPYLN(NS,LL)
          LT=LIT('NULL')
C         ***LEAVE SOURCE FILE STATUS AS 'INCOMPLETE PROGRAM LINE'
C         ***UNLESS CLOSING BRACKET IS FOUND
          JSFL=1
          IF(Q(LINE(LL),')')) JSFL=2
        ELSE
C         ***NO PROGRAM LINE USED, ADVANCE SOURCE FILE STATUS
C         ***TO 'PROGRAM LINE COMPLETE'
          JSFL=2
        ENDIF
      ENDIF
C
      IF(JSFL.EQ.1) THEN
C       ***NEXT LINE TREATED AS CONTINUATION PROGRAM LINE
        CALL COPYLN(NS,LL)
        LT=LIT('NULL')
        JSFL=1
        IF(Q(LINE(LL),')')) JSFL=2
      ENDIF
C
      IF(JSFL.EQ.2) THEN
C       ***PROGRAM LINE COMPLETE OR UNUSED, SO WRITE STANDARD
C       ***DECLARATIONS, STATUS BECOMES 'READY FOR TEXT'
        IF(JDEF.GT.0) CALL ADDDEC
        JSFL=3
      ENDIF
C
      IF(JSFL.EQ.3.AND..NOT.Q(LT,'NULL')) THEN
C       ***STATUS IS 'READY FOR TEXT' AND A LINE IS AVAILABLE
        IF(Q(LINE(IT),'$')) THEN
C         ***OPEN CONDITIONAL SECTION
          CALL ADDCON
C         ***STATUS BECOMES 'READY FOR CONDITIONALS'
          JSFL=4
        ELSE
C         ***COPY LINE TO GENERATED SOURCE FILE
          CALL COPYLN(NS,LL)
        ENDIF
      ENDIF
C
```

```
    IF(JSFL.EQ.4) THEN
C     ***STATUS IS 'READY FOR CONDITIONALS' AND LINES ARE
C     ***INTERPRETED AS SUCH
      IF(Q(LINE(IT),'$').AND.IT.LE.134) THEN
        IF(NDIGIT(LINE(IT+2)).GE.0) THEN
          IF(Q(LINE(IT+1),'H')) THEN
            CALL ADDHLP(IT)
          ELSEIF(Q(LINE(IT+1),'I')) THEN
            CALL ADDINF(IT)
          ENDIF
        ELSEIF(Q(K,'$E')) THEN
C         ***END 'CONDITIONAL TEXT' MODE
          JCT=0
        ELSEIF(.NOT.Q(K,'$')) THEN
C         ***SET 'CONDITIONAL TEXT' MODE
          JCT=1
          CALL ADDSEG(IT,IG)
        ENDIF
      ELSEIF(JCT.GE.1) THEN
C       ***COPY LINE INTO 'CONDITIONAL TEXT' SEGMENT
        CALL COPYLN(NS,LL)
      ELSE
C       ***GENERATE CONDITIONAL CALL TO NAMED ROUTINE
        CALL ADDSR(K,IG)
      ENDIF
    ENDIF
  ENDIF
C
C
C ***PROCESS GENERATED CONSTRUCTION PROCEDURE FILE   /6/
C
C
C ***OPEN FILE ON 'END' OR 'FILE' IF STATUS 'EMPTY'
  IF((Q(LT,'END').OR.Q(LT,'FILE')).AND.JPCFL.EQ.0) THEN
    CALL OPENPC(JDEF)
C   ***STATUS OF FILE BECOMES 'OPEN FOR SOURCE'
    JPCFL=1
  ENDIF
C
C ***ADD COMPILER STATEMENTS IF FILE 'OPEN FOR SOURCE' AT 'END'
C ***OR ON A BINARY FILE LINE
  IF(JPCFL.EQ.1) THEN
    IF(Q(LT,'END').OR.Q(K,'$B').OR.Q(K,'$BL')) THEN
      CALL COMPPC(JCMPL)
C     ***STATUS BECOMES 'OPEN FOR BINARY'
      JPCFL=2
    ENDIF
  ENDIF
C
C ***CLOSE FILE ON 'END'
  IF(Q(LT,'END')) THEN
    CALL ENDPC
C ***OR PROCESS A FILE LINE
  ELSEIF(Q(LT,'FILE')) THEN
    CALL FLSPC(K,IG,JDEF)
  ENDIF
C
C
C ***PROCESS GENERATED RUN PROCEDURE FILE   /7/
C
C ***OPEN FILE ON 'END' ,OR ON 'LIB', IF STATUS OF FILE 'EMPTY'
  IF((Q(LT,'END').OR.Q(LT,'LIB')).AND.JPRFL.EQ.0) THEN
    CALL OPENPR
C   ***STATUS OF FILE BECOMES 'OPEN'
    JPRFL=1
  ENDIF
C
C ***CLOSE FILE ON AN END LINE
  IF(Q(LT,'END')) THEN
    CALL ENDPR
C ***OR PROCESS A LIBRARY LINE
  ELSEIF(Q(LT,'LIB')) THEN
    CALL LIBPR(IG)
  ENDIF
C
C
 IF(.NOT.Q(LT,'END')) GO TO 100
C
 STOP
 END
C*** OF PROGRAM GEN
```

```
 SUBROUTINE ADDPRG (IG)
C
C GENERATES A PROGRAM STATEMENT AS SPECIFIED BY A '$P' LINE.
C
 COMMON/TXTLN/LINE(136)
 INTEGER T(40),JT(40)
 LOGICAL Q
C
 I=IG
 NT=0
C***CONVERT LINE INTO TOKENS FROM AFTER '$P'
 100 CONTINUE
   CALL LTOK(LINE,136,I)
   ITOK=I
   CALL LGAP(LINE,136,I)
   IGAP=I
   IF(ITOK.LE.136) THEN
     NT=NT+1
     CALL TXTLNN(ITOK- 1,IGAP,T(NT))
C    ***EXPAND 'I' OR 'O' TO 'INPUT' OR 'OUTPUT' UNLESS
C    ***IN FIRST (NAME) POSITION
     IF(NT.GT.1) THEN
       IF(Q(T(NT),'I')) T(NT)=LIT('INPUT')
       IF(Q(T(NT),'O')) T(NT)=LIT('OUTPUT')
     ENDIF
C    ***SET JT .GT.0 IF TOKEN NOT NUMERIC OR .GT.2 DIGITS
     JT(NT)=0
     CALL CNTDIG(LINE,136,ITOK-1,IGAP,ND)
     IF(ND.GT.2.OR.ND.LT.(IGAP-ITOK)) JT(NT)=1
   ENDIF
 IF(I.LT.136.AND.NT.LT.40) GO TO 100
C
C***ABANDON ATTEMPT IF NO FILE PARAMETERS
 IF(NT.LT.2) RETURN
C
C***START STATEMENT UP TO '('
 CALL TXTLNB(0)
 CALL TXTLNE(1,'PROGRAM')
 CALL TXTLNE(9,T(1))
 CALL TXTLNL(LL)
 LINE(LL+1)=LIT('(')
C
C***PROCESS FILE PARAMETERS
 DO 110 IT=2,NT
   IF(LL.GT.44) THEN
C    ***WRITE LINE AND START A CONTINUATION
     CALL COPYLN(0,LL)
     CALL TXTLNB(0)
     LINE(1)=LIT('+')
     LL=3
   ENDIF
C  ***COMMA SEPARATOR EXCEPT BEFORE FIRST PARAMETER
   LL=LL+1
   IF(IT.GE.3) LINE(LL)=LIT(',')
C  ***IF A NUMBER, PREFIX WITH 'TAPE'
   IF(JT(IT).LE.0) THEN
     CALL TXTLNE(LL,'TAPE')
     LL=LL+4
   ENDIF
C  ***OUTPUT PARAMETER
   CALL TXTLNE(LL,T(IT))
   CALL TXTLNL(LL)
C  ***IF PARAMETER AFTER FIRST AND PREVIOUS ONE NOT NUMERIC,
C  ***EQUATE THIS ONE TO ITS PREDECESSOR
   IF(IT.GT.2.AND.JT(IT-1).GT.0) THEN
     LL=LL+1
     LINE(LL)=LIT('=')
     CALL TXTLNE(LL,T(IT-1))
     CALL TXTLNL(LL)
   ENDIF
 110 CONTINUE
C
C***ADD CLOSING ')' AND WRITE LINE
 LL=LL+1
 LINE(LL)=LIT(')')
 CALL COPYLN(0,LL)
C
 RETURN
 END
C*** OF ADDPRG
```

```
 SUBROUTINE ADDSR (KA,IG)
C
C GENERATES A CONDITIONAL LINE CALLING A SUBROUTINE, AND A
C CONDITIONAL WRITE LINE TO DISPLAY ITS PARAMETER SPECIFICATION
C MESSAGE.
C
 COMMON/TXTLN/LINE(136)
 COMMON/LINENO/LNO
 INTEGER T(40),K(10)
 DATA (K(I),I=1,10)/'1','2','3','4','5','6','7','8','9','0'/
 LOGICAL Q
C
 I=IG
 NP=0
C***CONVERT REST OF LINE INTO TOKENS, K IS THE FIRST
 100 CONTINUE
   CALL LTOK(LINE,136,I)
   ITOK=I
   CALL LGAP(LINE,136,I)
   IGAP=I
   IF(ITOK.LE.136) THEN
     NP=NP+1
     CALL TXTLNW(ITOK-1,IGAP,T(NP))
   ENDIF
 IF(I.LT.136.AND.NP.LT.39) GO TO 100
C
C***GENERATE CONDITIONAL LINE SELECTING FOR SUBROUTINE NAME
 WRITE(6,1) LNO,KA
 1 FORMAT(I5,' ELSEIF(KA.EQ.LITT(',1H',A7,1H',')) THEN')
C***GENERATE CONDITIONAL WRITE AND FORMAT STATEMENT WITH MESSAGE
 WRITE(6,2) LNO+1,LNO
 2 FORMAT(I5,'   IF(JLST.GT.5) WRITE(IUMSG,',I5,')')
 IF(NP.GT.0) THEN
   CALL TXTLNL(LL)
   LL=MINO(LL,110)
   LINE(LL+1)=LIT(1H')
   LINE(LL+2)=LIT(')')
   WRITE(6,3) LNO+2,LNO,(LINE(I),I=IG,LL+2)
   3 FORMAT(I5,I8,' FORMAT(',1H',112A1)
 ELSE
   WRITE(6,4) LNO+2,LNO
   4 FORMAT(I5,I8,' FORMAT(',1H,' NO PARAMETERS',1H,')')
 ENDIF
 LNO=LNO+3
C
C***CONSTRUCT CONDITIONAL CALL TO SUBROUTINE
 CALL TXTLNB(0)
 CALL TXTLNE(3,'IF(JEX.GT.')
 CALL TXTLNE(13,'0) CALL ')
 CALL TXTLNE(21,KA)
C***ADD PARENTHESISED PARAMETER LIST IF REQUIRED
 IF(NP.GT.0) THEN
   IFP=1
   LL=29
   LINE(LL)=LIT('(')
C  ***LOOP OVER PARAMETERS
   DO 110 IP=1,NP
     IF(LL.GT.48) THEN
C      ***OUTPUT LINE AND START A CONTINUATION
       CALL COPYLN(0,LL)
       CALL TXTLNB(0)
       LINE(1)=LIT('+')
       LL=5
     ENDIF
C    ***PUT A COMMA BEFORE PARAMETERS EXCEPT THE FIRST
     IF(IP.GT.1) THEN
       LL=LL+1
       LINE(LL)=LIT(',')
     ENDIF
C    ***PUT PARAMETER
     M=T(IP)
     IF(Q(M,'R').OR.Q(M,'I').OR.Q(M,'K')) THEN
C      ***ONE OF STANDARD INPUT ARRAYS
       IFP=IFP+1
       IF(Q(M,'R')) LINE(LL+1)=LIT('R')
       IF(Q(M,'I')) LINE(LL+1)=LIT('I')
       IF(Q(M,'K')) LINE(LL+1)=LIT('K')
C      ***PUT ELEMENT NUMBER IN PARENTHESES
       LINE(LL+2)=LIT('(')
       LL=LL+2
       IF(IFP.GE.10) THEN
         LL=LL+1
         LINE(LL)=K(IFP/10)
       ENDIF
       LL=LL+1
       LINE(LL)=K(1+MOD((IFP-1),10))
       LL=LL+1
       LINE(LL)=LIT(')')
     ELSE
C      ***ACTUAL PARAMETER, COPY INTO STATEMENT
       CALL TXTLNE(LL,T(IP))
       CALL TXTLNL(LL)
     ENDIF
   110 CONTINUE
   LL=LL+1
C  ***CLOSE LIST
   LINE(LL)=LIT(')')
 ENDIF
C
C***OUTPUT LINE
 CALL COPYLN(0,LL)
C
 RETURN
 END
C*** OF ADDSR
C
C
C
 SUBROUTINE ADDSEG (IT,IG)
C
C GENERATES THE CONDITIONAL LINE STARTING A CONDITIONAL
C PROGRAM SEGMENT.
C
 COMMON/LINENO/LNO
C
 CALL TXTLNW(IT,IG,KCH)
 WRITE(6,1) LNO,KCH
 1 FORMAT(I5,' ELSEIF(KA.EQ.LITT(',1H',A10,1H',')) THEN')
 LNO=LNO+1
 RETURN
 END
C*** OF ADDSEG
C
C
C
 SUBROUTINE ADDHLP (IT)
C
C GENERATES A CONDITIONAL STATEMENT TO WRITE A 'HELP' MESSAGE.
C
 COMMON/LINENO/LNO
 COMMON/TXTLN/LINE(136)
C
 WRITE(6,1) LNO,LINE(IT+2),LNO
 1 FORMAT(I5,'   IF(JHLP.GT.',A1,')WRITE(IUMSG,',I5,')')
 CALL TXTLNL(LL)
 LL=MINO(LL,110)
 LINE(LL+1)=LIT(1H')
 LINE(LL+2)=LIT(')')
 WRITE(6,2) LNO+1,LNO,(LINE(I),I=IT+3,LL+2)
 2 FORMAT(I5,I8,' FORMAT(',1H',112A1)
 LNO=LNO+2
 RETURN
 END
C*** OF ADDHLP
C
C
C
 SUBROUTINE ADDINF (IT)
C
C GENERATES A CONDITIONAL STATEMENT TO WRITE AN 'INFORMATION'
C MESSAGE.
C
 COMMON/LINENO/LNO
 COMMON/TXTLN/LINE(136)
C
 WRITE(6,1) LNO,LINE(IT+2),LNO
 1 FORMAT(I5,'   IF(JINF.GT.',A1,')WRITE(IUMSG,',I5,')')
 CALL TXTLNL(LL)
 LL=MINO(LL,110)
 LINE(LL+1)=LIT(1H')
 LINE(LL+2)=LIT(')')
 WRITE(6,2) LNO+1,LNO,(LINE(I),I=IT+3,LL+2)
 2 FORMAT(I5,I8,' FORMAT(',1H',112A1)
 LNO=LNO+2
 RETURN
 END
C*** OF ADDINF
```

```
      SUBROUTINE ADDDEC
C
C WRITES THE STANDARD DECLARATIONS TO THE GENERATED SOURCE
C FILE.
C
      COMMON/LINENO/LNO
C
      WRITE(6,1) LNO,LNO+1,LNO+2,LNO+3,LNO+4,LNO+5,LNO+6,LNO+7,
     +  LNO+8,LNO+9,LNO+10,LNO+11
    1 FORMAT(I5,'C'/
     +         I5,' INTEGER I(40),IX(40),K(40),KX(40),LINE(136)'/
     +         I5,' REAL R(40),RX(40)'/
     +         I5,' COMMON/RUNVAR/I,K,LINE,R,LL,IT,KA,'/
     +         I5,'+  KBLANK,OLDTM,JEX,JINF,JHLP,JLST'/
     +         I5,' COMMON/ERR/JERR,IERR'/
     +         I5,' COMMON/SYSVAR/IUMSG,IULOG,IUSET,'/
     +         I5,'+  IUSTD,IURD,IUSEC,ISYS(10)'/
     +         I5,' DATA JSTART,IUREF/1,5/'/
     +         I5,' EQUIVALENCE (I(1),IX(1)),(K(1),KX(1))'/
     +         I5,' EQUIVALENCE (R(1),RX(1))'/
     +         I5,'C')
      LNO=LNO+12
      RETURN
      END
C*** OF ADDDEC
C
C
C
      SUBROUTINE ADDCON
C
C WRITES THE STANDARD STATEMENTS AT THE START OF THE
C CONDITIONALS SECTION OF THE SOURCE FILE.
C
      COMMON/LINENO/LNO
      WRITE(6,1) LNO,LNO+1,LNO+2,LNO+3,LNO+4,LNO+5
    1 FORMAT(I5,'C'/
     +         I5,' GO TO 9950'/
     +         I5,' 9900 CONTINUE'/
     +         I5,'C'/
     +         I5,' IF(LINE(IT).EQ.LITT(',3H'$',').OR.'
     +           ,'LINE(IT).EQ.LITT(',3H'*','')) THEN'/
     +         I5,'   NTOK=0')
      LNO=LNO+6
      RETURN
      END
C*** OF ADDCON
C
C
C
      SUBROUTINE ADDEND (JSFL)
C
C ENDS THE GENERATED SOURCE FILE.
C
      COMMON/LINENO/LNO
      IF(JSFL.LT.4) THEN
        WRITE(6,1) LNO
        1 FORMAT(I5,'C')
        LNO=LNO+1
      ENDIF
      RETURN
      END
C*** OF ADDEND
C
C
C
      SUBROUTINE COPYLN (NS,LL)
C
C OUTPUTS LINE FROM CHARACTER AFTER NS TO UNIT 6, UP TO LINE
C LENGTH LIMIT AND SUPRESSING TRAILING BLANKS, AND RETURNS
C POINTER TO LAST CHARACTER OF INPUT LINE AS LL.
C
      COMMON/LINENO/LNO
      COMMON/TXTLN/LINE(136)
C
      CALL TXTLNL(LL)
      WRITE(6,1) LNO,(LINE(I),I=NS+1,MINO(LL,131+NS))
      1 FORMAT(I5,131A1)
      LNO=LNO+1
      RETURN
      END
C*** OF COPYLN
      SUBROUTINE OPENPC (JDEF)
C
C WRITES THE OPENING COMMANDS TO FILE PC.
C
      WRITE(7,1)
      1 FORMAT('REWIND,BS'/'REWIND,S'/'COPY,S,BS')
      IF(JDEF.GT.0) WRITE(7,2)
      2 FORMAT('GET,SRUN'/'COPY,SRUN,BS'/
     +         'GET,SLCON'/'COPY,SLCON,BS')
      RETURN
      END
C*** OF OPENPC
C
C
C
      SUBROUTINE COMPPC (JCMPL)
C
C WRITES THE COMPILER COMMAND TO FILE PC.
C
      WRITE(7,1)
      1 FORMAT('*'/'RETURN,T'/'PACK,BS,T'/'REWIND,BS'/
     +         'REWIND,T'/'*')
      IF(JCMPL.GE.0) WRITE(7,2)
      2 FORMAT('MNF5,B=BS,K,I=T'/'*')
      RETURN
      END
C*** OF COMPPC
C
C
C
      SUBROUTINE ENDPC
C
C WRITES THE CLOSING COMMANDS TO FILE PC.
C
      WRITE(7,1)
      1 FORMAT('PACK,BS'/'REWIND,BS'/'REWIND,S'/'REWIND,T')
      RETURN
      END
C*** OF ENDPC
C
C
C
      SUBROUTINE FLSPC (K,IG,JDEF)
C
C WRITES THE COMMANDS TO FILE PC TO ACCESS AND APPEND THE
C FILES CALLED FOR IN A 'FILE' LINE.
C
      COMMON/TXTLN/LINE(136)
      INTEGER NAME(20)
      LOGICAL Q
C
      I=IG
      IN=0
C***CONVERT LINE AFTER THE SPECIFIER K TO TOKENS
  100 CONTINUE
        CALL LTOK(LINE,136,I)
        ITOK=I
        CALL LGAP(LINE,136,I)
        IGAP=I
        IF(ITOK.LE.136) THEN
          IN=IN+1
          CALL TXTLNW(ITOK-1,IGAP,NAME(IN))
        ENDIF
C  ***DELETE TOKEN IF IT CALLS FOR A FILE ADDED AS STANDARD
        IF(JDEF.GT.0.AND.Q(NAME(IN),'SLCON')) IN=IN-1
      IF(I.LT.136.AND.IN.LT.20) GO TO 100
C
C***RETURN IF THERE ARE NO TOKENS AFTER SPECIFIER
      IF(IN.LE.0) RETURN
C
C***LOOP OVER TOKENS
      DO 110 IFL=1,IN
        KA=LIT('GET')
C  ***SUBSTITUTE REWIND FOR GET IF A LOCAL FILE
        IF(Q(K,'$SL').OR.Q(K,'$BL')) KA=LIT('REWIND')
        KB=NAME(IFL)
C  ***GENERATE GET OR REWIND COMMAND AND COPY COMMAND
        CALL CMDAB(7,KA,KB)
        CALL CMDABC(7,'COPY',KB,'BS')
  110 CONTINUE
C
      RETURN
      END
C*** OF FLSPC
```

```
 SUBROUTINE OPENPR
C
C OPENS THE PR FILE. DUMMY.
C
 COMMON/D/ID
 ID=0
 RETURN
 END
C*** OF OPENPR (DUMMY)
C
C
C
 SUBROUTINE LIBPR (IG)
C
C GENERATES STATEMENTS TO FILE PR TO CAUSE USE OF SPECIFIED
C LIBRARY. DUMMY.
C
 COMMON/D/ID
 ID=IG
 RETURN
 END
C*** OF LIBPR (DUMMY)
C
C
C
 SUBROUTINE ENDPR
C
C ENDS FILE PR. DUMMY.
C
 COMMON/D/ID
 ID=0
 RETURN
 END
C*** OF ENDPR (DUMMY)
C
C
C
 SUBROUTINE CMPSET (IG,JCMPL)
C
C SETS PARAMETERS CONTROLLING COMPILER TO BE USED FROM
C INPUT LINE.
C
 COMMON/TXTLN/LINE(136)
 IT=IG
 CALL LTOK(LINE,136,IT)
 IF(IT.LE.136) THEN
   IF(LINE(IT).EQ.LIT('N')) JCMPL=-1
 ENDIF
 RETURN
 END
C*** OF CMPSET (DUMMY)
C
C
C
 SUBROUTINE CMDWRT (IU,KP,NP)
C
C WRITES THE CONTENTS OF KP(NP) TO UNIT IU AS A COMMAND
C STATEMENT WITH ',' SEPARATORS AND NO EMBEDDED BLANKS.
C IF NP=1 A TRAILING '.' IS ADDED. IF AN ITEM ENDS IN
C '=' THE FOLLOWING ',' IS SUPRESSED. IF IT IS '=' THE
C PRECEDING ',' IS ALSO SUPRESSED.
C
 COMMON/TXTLN/LINE(136)
 INTEGER KP(NP)
 LOGICAL Q
C
 IF(NP.LE.0) RETURN
C
 CALL TXTLNB(0)
 CALL TXTLNE(0,KP(1))
 IF(NP.EQ.1) THEN
C  ***TERMINATE A SINGLE WORD WITH '.'
   CALL TXTLNL(LL)
   LINE(LL+1)=LIT('.')
 ELSE
C  ***LOOP OVER WORDS TO FORM COMMAND
   DO 100 IP=2,NP
     CALL TXTLNL(LL)
C    ***PREFIX WITH A COMMA SEPARATOR EXCEPT IN SPECIAL CASES
     IF(.NOT.(Q(LINE(MAXO(1,LL)),'=').OR.Q(KP(IP),'='))) THEN
       LL=LL+1
       LINE(LL)=LIT(',')
     ENDIF
C    ***ADD WORD
     CALL TXTLNE(LL,KP(IP))
   100 CONTINUE
 ENDIF
C***OUTPUT GENERATED COMMAND LINE
 CALL TXTLNP(IU,0)
 RETURN
 END
C*** OF CMDWRT
 SUBROUTINE CMDAB (IU,KA,KB)
C
C WRITES A COMMAND STATEMENT TO IU FORMED FROM TWO WORDS KA,KB.
C
 INTEGER KP(2)
 KP(1)=KA
 KP(2)=KB
 CALL CMDWRT(IU,KP,2)
 RETURN
 END
C*** OF CMDAB
C
C
C
 SUBROUTINE CMDABC (IU,KA,KB,KC)
C
C WRITES A COMMAND STATEMENT TO IU FORMED OF THREE WORDS.
C
 INTEGER KP(3)
 KP(1)=KA
 KP(2)=KB
 KP(3)=KC
 CALL CMDWRT(IU,KP,3)
 RETURN
 END
C*** OF CMDABC
C
C
C
 SUBROUTINE TXTLNA (N,KLINE,LKL)
C
C 'ADD'. COPIES ANY NON BLANK CHARACTERS IN KLINE(LKL) INTO
C LINE STARTING WITH POSITION N+1. 'INSERT' IS THE EQUIVALENT
C COPYING ALL CHARACTERS.
C
 COMMON/TXTLN/LINE(136)
 DATA KBLANK/1H /
 INTEGER KLINE(LKL)
C
 DO 100 I=1,LKL
   J=I+N
   IF(J.GT.0.AND.J.LE.136) THEN
     IF(KLINE(I).NE.KBLANK) LINE(J)=KLINE(I)
   ENDIF
 100 CONTINUE
 RETURN
 END
C*** OF TXTLNA
C
C
C
 SUBROUTINE TXTLNB (N)
C
C 'BLANK'. SETS LINE TO BLANK AFTER THE FIRST N CHARACTERS.
C
 COMMON/TXTLN/LINE(136)
 DATA KBLANK/1H /
C
 DO 100 I=N+1,136
 100 IF (I.GT.0.AND.I.LE.136) LINE(I)=KBLANK
 RETURN
 END
C*** OF TXTLNB
C
C
C
 SUBROUTINE TXTLNE (N,KCH)
C
C 'ENTER'. COPIES ANY NON BLANK CHARACTERS IN A WORD KCH INTO
C  LINE STARTING AT POSITION N+1.
C
 INTEGER KS(10)
C
 CALL WRDEXP(KCH,KS,10)
 CALL TXTLNA(N,KS,10)
 RETURN
 END
C*** OF TXTLNE
```

```
      SUBROUTINE TXTLNL (N)
C
C 'LENGTH'. RETURNS THE POSITION OF THE LAST NON BLANK
C  CHARACTER IN LINE.
C
      COMMON/TXTLN/LINE(136)
      DATA KBLANK/1H /
C
      N=137
  100 N=N-1
        IF(N.LE.0) RETURN
        IF(LINE(N).NE.KBLANK) RETURN
      GO TO 100
      END
C*** OF TXTLNL
C
C
C
      SUBROUTINE TXTLNP (IU,JP)
C
C 'PUT'. WRITES LINE TO UNIT IU. IF JP IS NON ZERO, A
C  CHARACTER IS PREFIXED TO THE LINE. 1=BLANK 2=ONE 3=PLUS
C
      COMMON/TXTLN/LINE(136)
      DATA KBLANK,KONE,KPLUS/1H ,1H1,1H+/
C
      IF(IU.LE.0.OR.IU.GE.100) RETURN
C
      CALL TXTLNL(N)
      IF(JP.EQ.0) THEN
        IF(N.LE.0) WRITE(IU,1)
        IF(N.GT.0) WRITE(IU,1) (LINE(I),I=1,N)
        1 FORMAT(136A1)
      ELSE
        KP=KBLANK
        IF(JP.EQ.2) KP=KONE
        IF(JP.EQ.3) KP=KPLUS
        IF(N.LE.0) WRITE(IU,2) KP
        IF(N.GT.0) WRITE(IU,2) KP,(LINE(I),I=1,MINO(N,135))
        2 FORMAT(136A1)
      ENDIF
C
      RETURN
      END
C*** OF TXTLNP
C
C
C
      SUBROUTINE TXTLNW (N,NL,KCH)
C
C 'WORD'. TAKES CHARACTERS IN LINE FROM N+1 TO NL-1 OR TO A
C MAXIMUM OF 10 AND PACKS THEM INTO KCH WITH BLANK FILL.
C
      COMMON/TXTLN/LINE(136)
      DATA KBLANK/1H /
C
      DO 100 I=N+1,N+10
        KCH=SHIFT(KCH,6).AND..NOT.77B
        L=KBLANK
        IF(I.GE.1.AND.I.LE.136.AND.I.LT.NL)
     +    L=SHIFT(LINE(I),-54)
        KCH=KCH.OR.(L.AND.77B)
  100 CONTINUE
      RETURN
      END
C*** OF TXTLNW
C
C
C
      SUBROUTINE WRDEXP (KWRD,KCHS,LCHS)
C
C EXPANDS A CDC WORD INTO SEPARATE CHARACTERS IN KCHS(LCHS).
C
      DATA KBLANK/1H /
      INTEGER KCHS(LCHS)
C
      K=KWRD
      MK=77000000000000000000B
      DO 100 I=1,10
        IF(I.LE.LCHS) KCHS(I)=(K.AND.MK).OR.(KBLANK.AND..NOT.MK)
        K=SHIFT(K,6)
  100 CONTINUE
      IF(LCHS.GT.10) THEN
        DO 110 I=11,LCHS
        110 KCHS(I)=KBLANK
      ENDIF
      RETURN
      END
C*** OF WRDEXP
      SUBROUTINE CNTDIG (K,LK,N,NL,ND)
C
C RETURNS THE NUMBER OF CHARACTERS ND THAT ARE DIGITS BETWEEN
C N AND NL EXCLUSIVE (NL.GT.N) IN K(LK).
C
      INTEGER K(LK)
C
      ND=0
      IF((NL-N).LT.2) RETURN
      DO 100 I=N+1,NL-1
        ID=-1
        IF(I.GE.1.AND.I.LE.LK) ID=NDIGIT(K(I))
        IF(ID.GE.0) ND=ND+1
  100 CONTINUE
      RETURN
      END
C*** OF CNTDIG
C
C
C
      SUBROUTINE LTOK (K,L,N)
C
C LOCATES THE NEXT NON-SPACING CHARACTER IN A CHARACTER
C ARRAY K(L) AFTER POSITION N, SETTING N TO THE NEW
C POSITION. USED BY CONLIN TO FIND THE START OF THE NEXT
C TOKEN.
C
      INTEGER K(L)
      DATA KBLANK,KCOMMA/1H ,1H,/
C
  100 N=N+1
        IF(N.GT.L) RETURN
        IF(K(N).NE.KBLANK.AND.K(N).NE.KCOMMA) RETURN
      GO TO 100
C
      END
C*** OF LTOK
C
C
C
      SUBROUTINE LGAP (K,L,N)
C
C FINDS THE NEXT SPACING CHARACTER IN ARRAY K(L) AFTER
C POSITION N, SETTING N TO THE NEW POSITION. USED BY CONLIN
C TO FIND THE END OF A TOKEN.
C
      INTEGER K(L)
      DATA KBLANK,KCOMMA/1H ,1H,/
C
  100 N=N+1
        IF(N.GT.L) RETURN
        IF(K(N).EQ.KBLANK.OR.K(N).EQ.KCOMMA) RETURN
      GO TO 100
C
      END
C*** OF LGAP
C
C
C
      FUNCTION LIT (KWRD)
C
C RETURNS THE VALUE OF ITS PARAMETER. TO PERMIT GENERAL USE
C OF CHARACTER LITERALS.
C
      LIT=KWRD
      RETURN
      END
C*** OF FUNCTION LIT
C
C
C
      FUNCTION NDIGIT (KCH)
C
C RETURNS THE INTEGER VALUE OF FIRST DIGIT OF KCH. IF CHARACTER
C NOT A DIGIT, RETURNS A VALUE .LT.0 UNIQUE TO CHARACTER.
C
      N=SHIFT(KCH,-54).AND.77B
      N=N-27
      IF(N.GT.9) N=N-64
      NDIGIT=N
      RETURN
      END
C*** OF FUNCTION NDIGIT
C
C
C
      LOGICAL FUNCTION Q (KA,KB)
C
C RETURNS TRUE IF KA.EQ.KB
C
      Q=KA.EQ.KB
      RETURN
      END
C*** OF LOGICAL FUNCTION Q
```

B1
Sample Input: to Plot a Star Field

This is a simple, annotated, example of the terminal input required to generate a grid of stars, add noise, and filter, and then plot the image. System prompts and responses are not shown: the responses are only the times for plotting and plot file lengths, program execution times, and plot costs. We assume the image generation and plotting programs have been compiled as BSTR, BPL.

Generate the image...

```
BSTR                          :run the image generation program
$TERM                         :initialise for use from terminal
GRDCLR 4 4                    :set up an empty 16 element grid of stars
GRDDEF .2 .8 .2 .8            :define dimensions of grid
GRDSET 3 10 0 -.1             :enter brightness into element 3 as an arithmetic
                               series 0 to -1.5
GRDSET 3 4                    :convert element 3 to antilog, so it is in steps of
                               ¼ magnitude
FLSET 0 1 101 101             :set image size in pixels
STRGEN 1 7 .02                :generate stars to file 7 with sigma 0.02
NOISE 7 8 0 .05               :copy from 7 to 8 adding noise amplitude 0.05
FLTXY 8 9 13 2                :filter from 8 to 9, gaussian with sigma 2 pixels, 13
                               pixels wide
$END                          :stop the program
```

...and plot it.

```
-MICFILM(L=BPL)               :execute the system microfilm procedure, which
                               will run the plotting program
$TERM                         :initialise for use from a terminal
PLOTZ 89 89 -100 1000 1 305 9 :plot a continuous tone image from file 9. Image
                               size is 89 by 89. Plotting range is -0.1 to 0.9.
                               Identity numbers are 1, 305.
PLOTC 89 89 0 20 2 305 9      :and also a contour map, 20 setting the lowest
                               contour level (at 0.032)
ENPLOT                        :call microfilm system routine to close the plot
$END                          :stop the program
```

B2
Sample Input:
for Generator Program

This is a sample input file to the command interpreter generator program, followed by the generated procedure file and the start of the command interpreter produced.

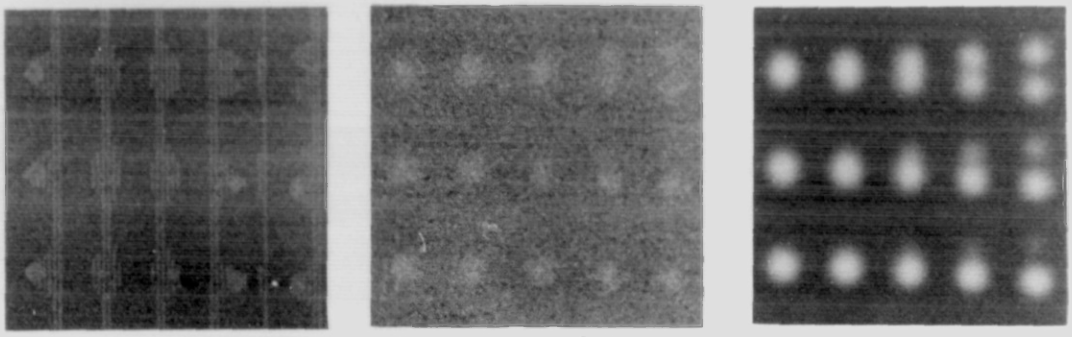
```
$P TEST I 5 0 6 7 8 9 10
 REAL ST(300),DATA(300)
 INTEGER TEXT(1)
 COMMON B(5000)
 NSTEL=4
 NR=2
 NC=3
$
STRCLR NS
STRSET ST 300 NSTEL NS RX 40
STRMOD ST NS NSTEL RX 40
STRLST I ST 300 NSTEL NS
STRGEN I I ST NS NSTEL DATA 300 R R I
STRFLD NS R R R R
STRINS ST 300 NSTEL NS R I
STRDBL ST 300 4 NS R R R I
NOISE I I I DATA 300 R R R
FLLST I I I I R
FLSTAT DATA 300 I R R I I   R
FLSUM DATA 300 B 5000 I I I R R   R I
GRDCLR  ST 300 NSTEL NS NR NC   I I R
GRDDEF  ST 300 NSTEL    NR NC   R R R R R
GRDSET  ST 300 NSTEL    NR NC   I I R R R
GRDRND  ST 300 NSTEL    NR NC   I
```

```
GRDDBS  ST 300 NSTEL    NR NC   I
$NSTEL
 NSTEL=I(2)
$E
*
FLTXY B 5000 I I I R I  I R I
FLT   B 5000 I I I
*
$FLDATA
   CALL FLDATA(I(2),I(3),IATYPE,LL,NL)
   WRITE(6,*) IATYPE,LL,NL
$E
FLSET I I I I
FLSIZE  I I I
*
$S SSTRST SFLDMY SRDG SGRD SSTRGN SNOISE SFLT

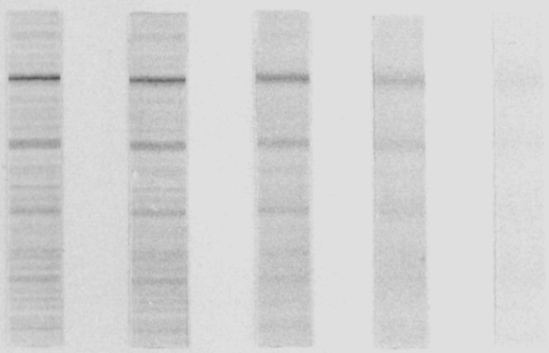

        REWIND,BS
        REWIND,S
        COPY,S,BS
        GET,SRUN
        COPY,SRUN,BS
        GET,SLCON
        COPY,SLCON,BS
        GET,SSTRST
        COPY,SSTRST,BS
        GET,SFLDMY
        COPY,SFLDMY,BS
        GET,SRDG
        COPY,SRDG,BS
        GET,SGRD
        COPY,SGRD,BS
        GET,SSTRGN
        COPY,SSTRGN,BS
        GET,SNOISE
        COPY,SNOISE,BS
        GET,SFLT
        COPY,SFLT,BS
        *
        RETURN,T
        PACK,BS,T
        REWIND,BS
        REWIND,T
        *
        MNF5,B=BS,K,I=T
        *
        PACK,BS
        REWIND,BS
        REWIND,S
        REWIND,T
```

```
10000 PROGRAM TEST(INPUT,TAPE5=INPUT,OUTPUT,TAPE6=OUTPUT
10001+   ,TAPE7,TAPE8,TAPE9,TAPE10)
10002C
10003 INTEGER I(40),IX(40),K(40),KX(40),LINE(136)
10004 REAL R(40),RX(40)
10005 COMMON/RUNVAR/I,K,LINE,R,LL,IT,KA,
10006+   KBLANK,OLDTM,JEX,JINF,JHLP,JLST
10007 COMMON/ERR/JERR,IERR
10008 COMMON/SYSVAR/IUMSG,IULOG,IUSET,
10009+   IUSTD,IURD,IUSEC,ISYS(10)
10010 DATA JSTART,IUREF/1,5/
10011 EQUIVALENCE (I(1),IX(1)),(K(1),KX(1))
10012 EQUIVALENCE (R(1),RX(1))
10013C
10014 REAL ST(300),DATA(300)
10015 INTEGER TEXT(1)
10016 COMMON B(5000)
10017 NSTEL=4
10018 NR=2
10019 NC=3
10020C
10021 GO TO 9950
10022 9900 CONTINUE
10023C
10024 IF(LINE(IT).EQ.LITT('$').OR.LINE(IT).EQ.LITT('*')) THEN
10025   NTOK=0
10026 ELSEIF(KA.EQ.LITT('STRCLR ')) THEN
10027   IF(JLST.GT.5) WRITE(IUMSG,10026)
10028   10026 FORMAT(' NS')
10029   IF(JEX.GT.0) CALL STRCLR (NS)
10030 ELSEIF(KA.EQ.LITT('STRSET ')) THEN
10031   IF(JLST.GT.5) WRITE(IUMSG,10030)
10032   10030 FORMAT(' ST 300 NSTEL NS RX 40')
10033   IF(JEX.GT.0) CALL STRSET (ST,300,NSTEL,NS,RX,40)
10034 ELSEIF(KA.EQ.LITT('STRMOD ')) THEN
10035   IF(JLST.GT.5) WRITE(IUMSG,10034)
10036   10034 FORMAT(' ST NS NSTEL RX 40')
10037   IF(JEX.GT.0) CALL STRMOD (ST,NS,NSTEL,RX,40)
10038 ELSEIF(KA.EQ.LITT('STRLST ')) THEN
10039   IF(JLST.GT.5) WRITE(IUMSG,10038)
10040   10038 FORMAT(' I ST 300 NSTEL NS')
10041   IF(JEX.GT.0) CALL STRLST (I(2),ST,300,NSTEL,NS)
10042 ELSEIF(KA.EQ.LITT('STRGEN ')) THEN
10043   IF(JLST.GT.5) WRITE(IUMSG,10042)
10044   10042 FORMAT(' I I ST NS NSTEL DATA 300 R R I')
10045   IF(JEX.GT.0) CALL STRGEN (I(2),I(3),ST,NS,NSTEL
10046+     ,DATA,300,R(4),R(5),I(6))
10047 ELSEIF(KA.EQ.LITT('STRFLD ')) THEN
10048   IF(JLST.GT.5) WRITE(IUMSG,10047)
10049   10047 FORMAT(' NS R R R R')
10050   IF(JEX.GT.0) CALL STRFLD (NS,R(2),R(3),R(4),R(5))
10051 ELSEIF(KA.EQ.LITT('STRINS ')) THEN
10052   IF(JLST.GT.5) WRITE(IUMSG,10051)
10053   10051 FORMAT(' ST 300 NSTEL NS R I')
10054   IF(JEX.GT.0) CALL STRINS (ST,300,NSTEL,NS,R(2)
10055+     ,I(3))
```

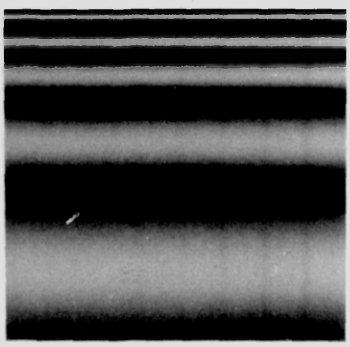
C: THE PLATES



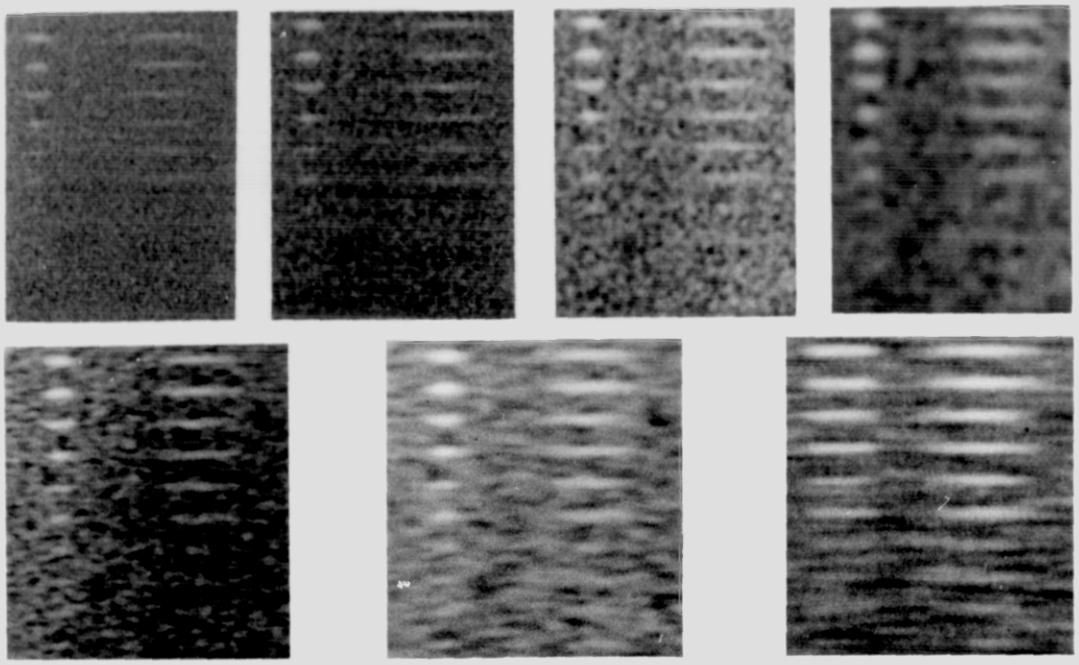
Pl.2.2a Effect of jitter on continuous tone plotting.
Left: image plotted at low level (peak 2.5% of saturation) without jitter.
Centre: as left, but with jitter.
Right: image plotted at a high level.



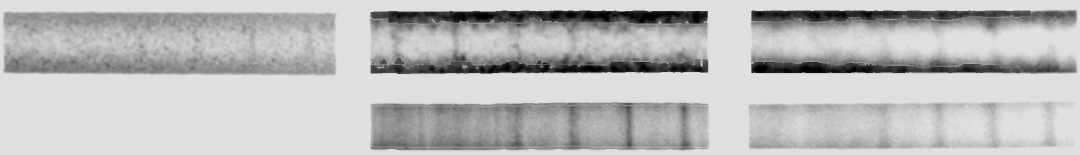
Pl.3.1a One dimensional peaks in noise.
Each plot has 4 lines, matched filter Sn 2.8, 4, 5.6, 8.
Filter gaussian, Wf/Ws (l to r): ¼, ½, **1**, 2, 4.



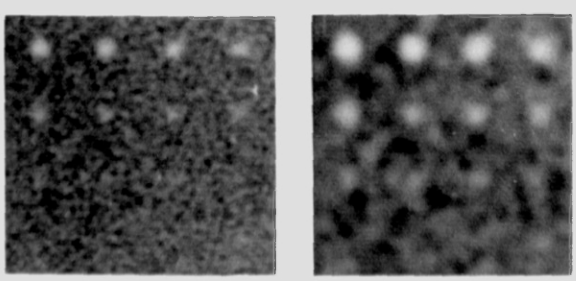
Pl.3.1b Line depth needed for visibility.
Gaussian transverse and line profiles, widths ratio: ½, 1, 2, 4, 8.
Percent depth in original: 2, 2.8, 4, 5.6, 8, 11, 16.



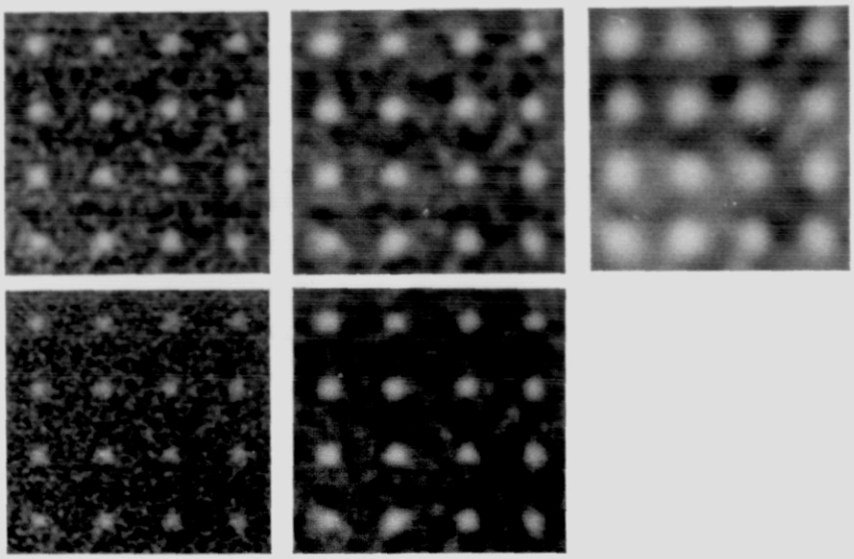
Pl.3.1c Emission lines in noise.
Gaussian transverse and line profiles, widths ratio: 3, 10.
Matched filter Sn (in each frame): 2 each 3, 5, 8, 12.5, 20.
Upper (l to r): no filter; filter gaussian Wf/Wl=½, 1, 2.
Lower (l to r): non-circular filter half matched widths of shorter lines, matched to shorter lines, approximately matched to longer lines.



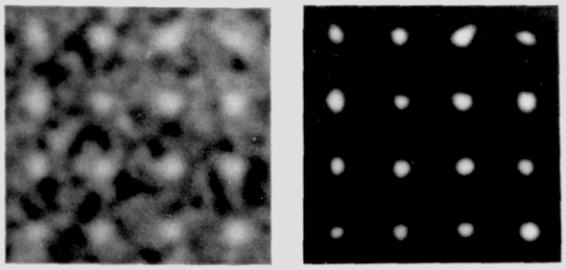
Pl.3.1d Absorption lines with noise.
Gaussian transverse and line profiles, widths ratio 3. Noise approximates to photon noise.
Matched filter Sn (in each frame): 2 each 3, 5, 8.
Upper (l to r): no filter; filter gaussian (circularly symmetric) Wf/Wl=½, 1.
Lower (l to r): as above, but with nearly uniform transverse weighting.



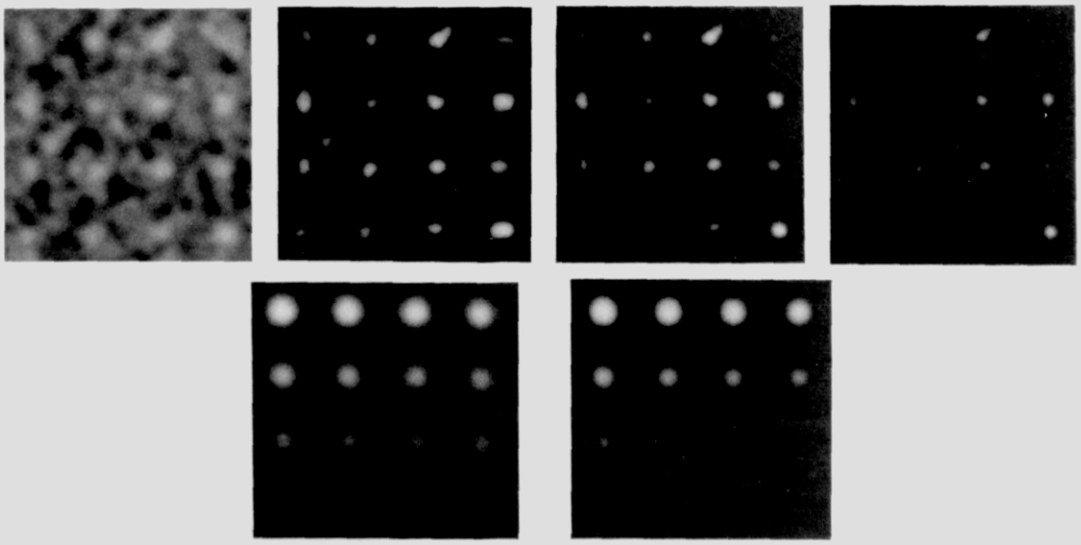
Pl.3.2a Star series in noise.
Brightness at ¼ magnitude intervals. Matched filter Sn of brightest star 32.
Left: no filter (Ws 2 pixels). Right: matched filter.



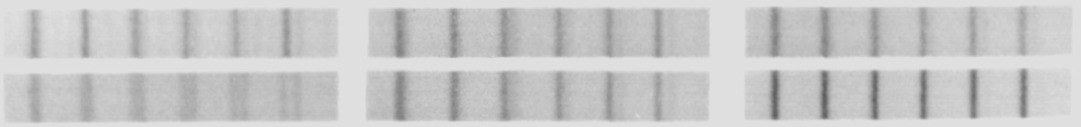
Pl.3.2b Stars in noise.
Matched filter Sn 12.5.
Upper (l to r): gaussian filter Wf/Ws= ½, 1, 2.
Lower (l to r): no filter (Ws 2 pixels), square filter (side 2.5Ws).



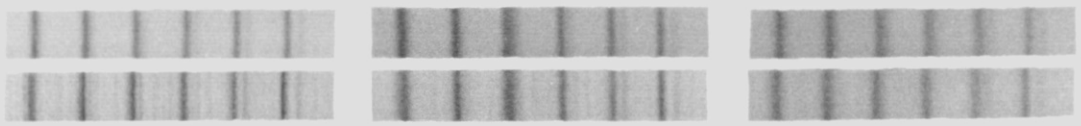
Pl.3.2c Suppressing a noisy background.
Stars Sn=8, matched filter.
Left: linear plot. Right: with background suppressed.



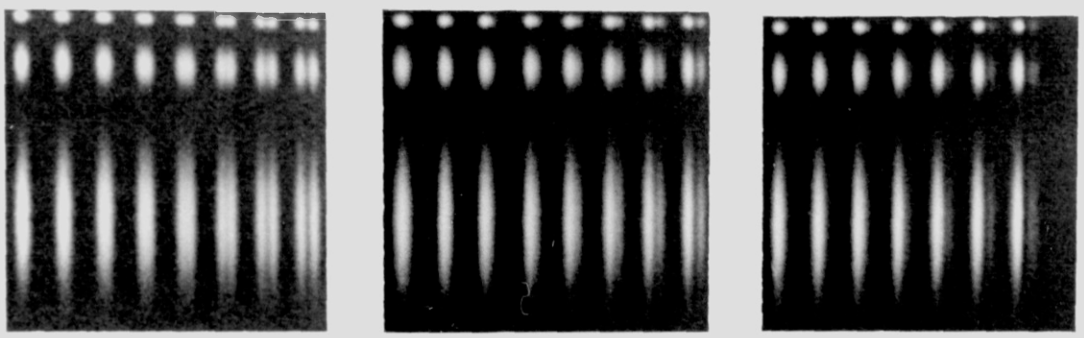
Pl.3.2d Disadvantages of suppressing a noisy background.
Upper: stars Sn=5, matched filter. Linear and with different threshold settings.
Lower: star series at ¼ magnitude intervals without noise, linear and with suppressed background.



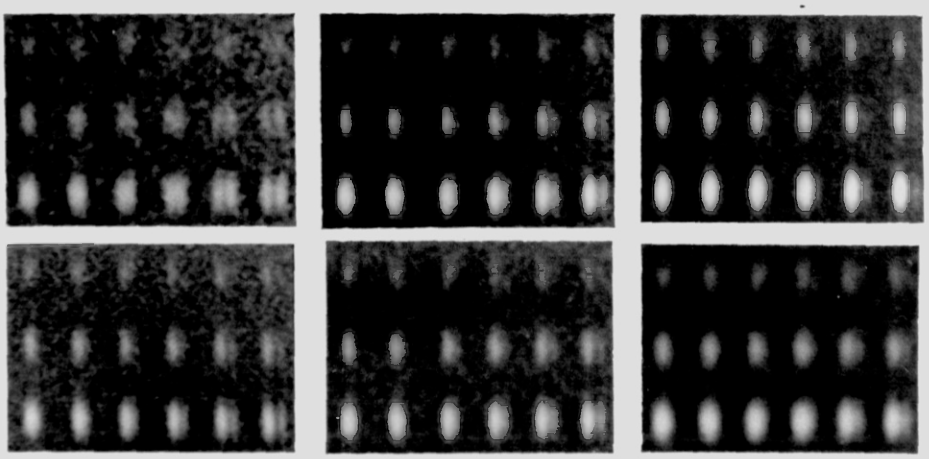
Filter gaussian Wf/Wl=½. Separation decreasing by factors of fourth root of 2 (1.19) from 4Wl, or as indicated.
Upper: ratio 3, matched filter Sn of weaker component (l to r): 5, 12.5, very high.
Lower: without noise, ratio (l to r): 1, 3, 9 (sep from 4.8Wl).



Ratio 3. Separation decreasing by factors of fourth root of 2 (1.19) from 4Wl.
Filter (l to r): none (Wl 2 pixels), gaussian Wf/Wl=½,1.
Upper: no noise. Lower: weak component matched filter Sn 5.

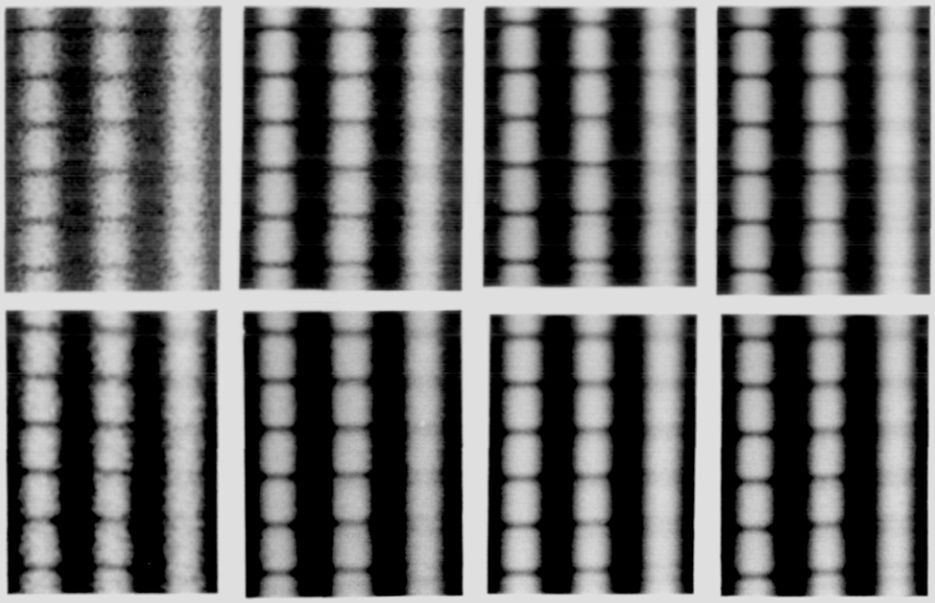


No noise, no filter. Ratios of transverse to line widths: 1, 3, 10. Separation reducing by factors of fourth root of 2 (1.19).
Ratio, max sep (l to r): 1, 4Wl; 3, 4Wl; 9, 4.8Wl.

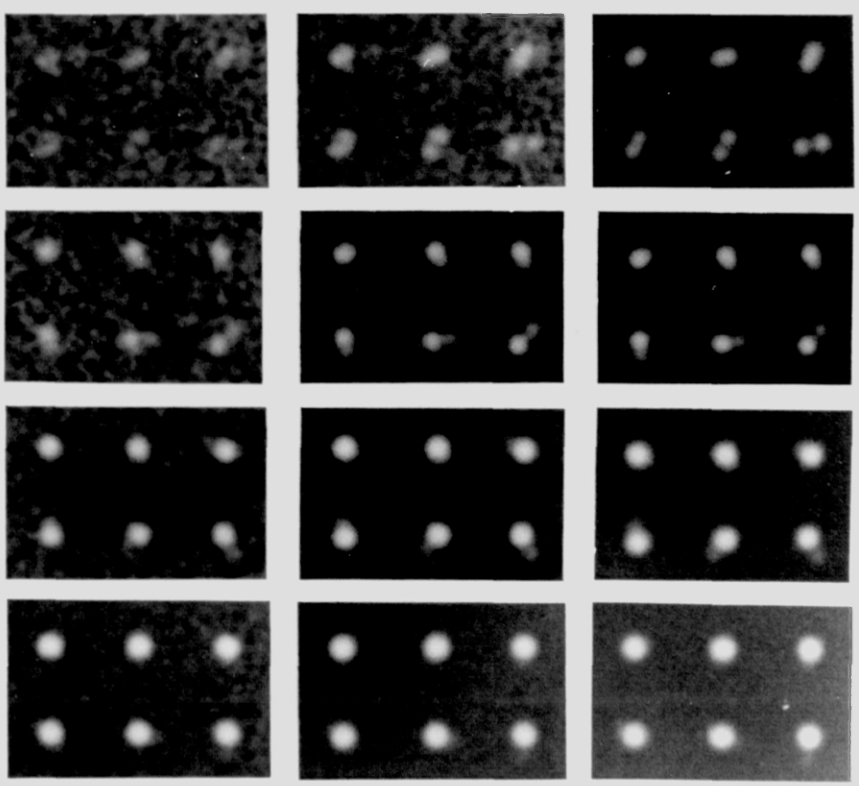


Ratio of transverse to line widths 3. Separation reducing by factors of fourth root of 2 (1.19) from 4Wl, except as indicated.
Weak component matched filter Sn (in each frame): 5, 12.5, 31.
Upper: gaussian filter Wf/Wl=½, ratio (l to r): 1, 3, 9 (max sep 4.8Wl).
Lower: ratio 3; filter none (Wl 1.4 pixels), gaussian Wf/Wl=½, 1.

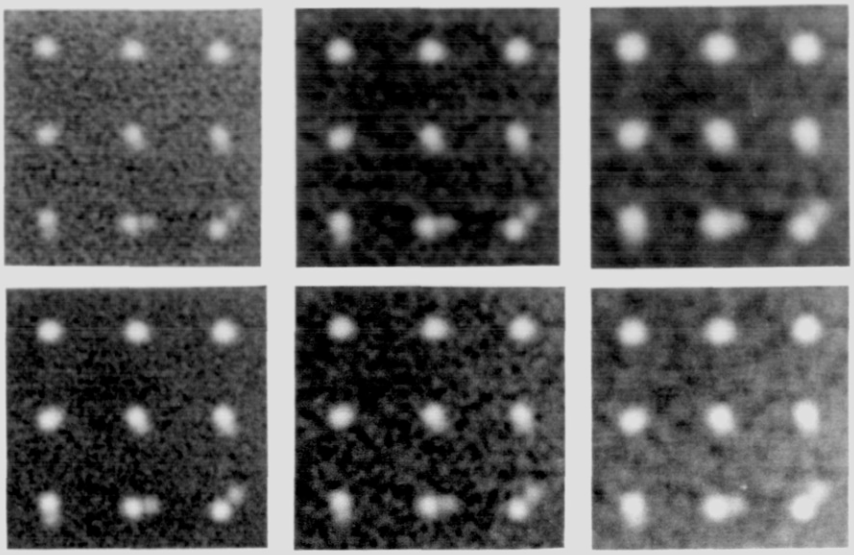
Pl.3.3a(i) Appearance of double lines.



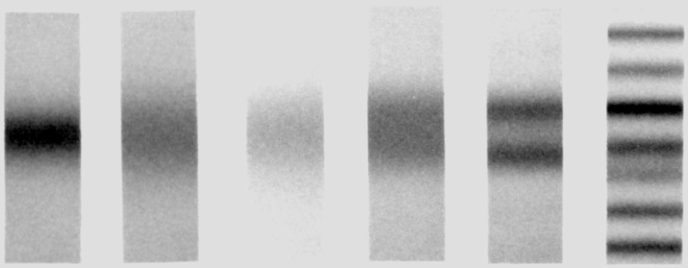
Pl.3.3a(ii) Appearance of double lines (cont).
Ratio of transverse to line widths: 3. Separation reducing by factors of fourth root of 2 (1.19) from 4Wl. Ratios (in each frame): 1, 3, 9. Noise approximates to photon noise.
Weak component matched filter Sn (l to r): 5, 12.5, 31, very high.
Upper: unfiltered (Wl 1.4 pixels). Lower: gaussian filter Wf/Wl=½.



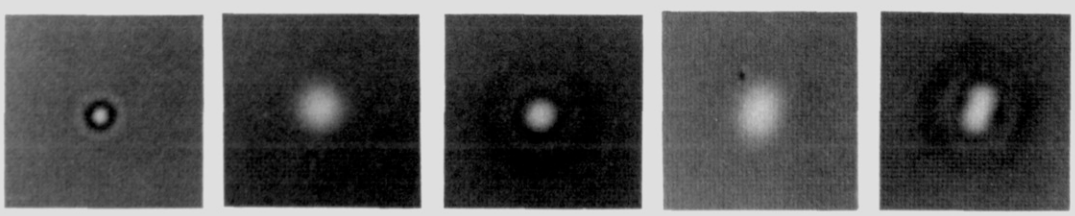
Pl.3.4a(i) Appearance of double stars.
Filter gaussian Wf/Ws=½. Separation reducing by factors of fourth root of 2 (1.19) from 4Ws.
Weak component matched filter Sn (l to r): 5, 12.5, very high.
Ratio (t to b): 1, 3, 9, 27.



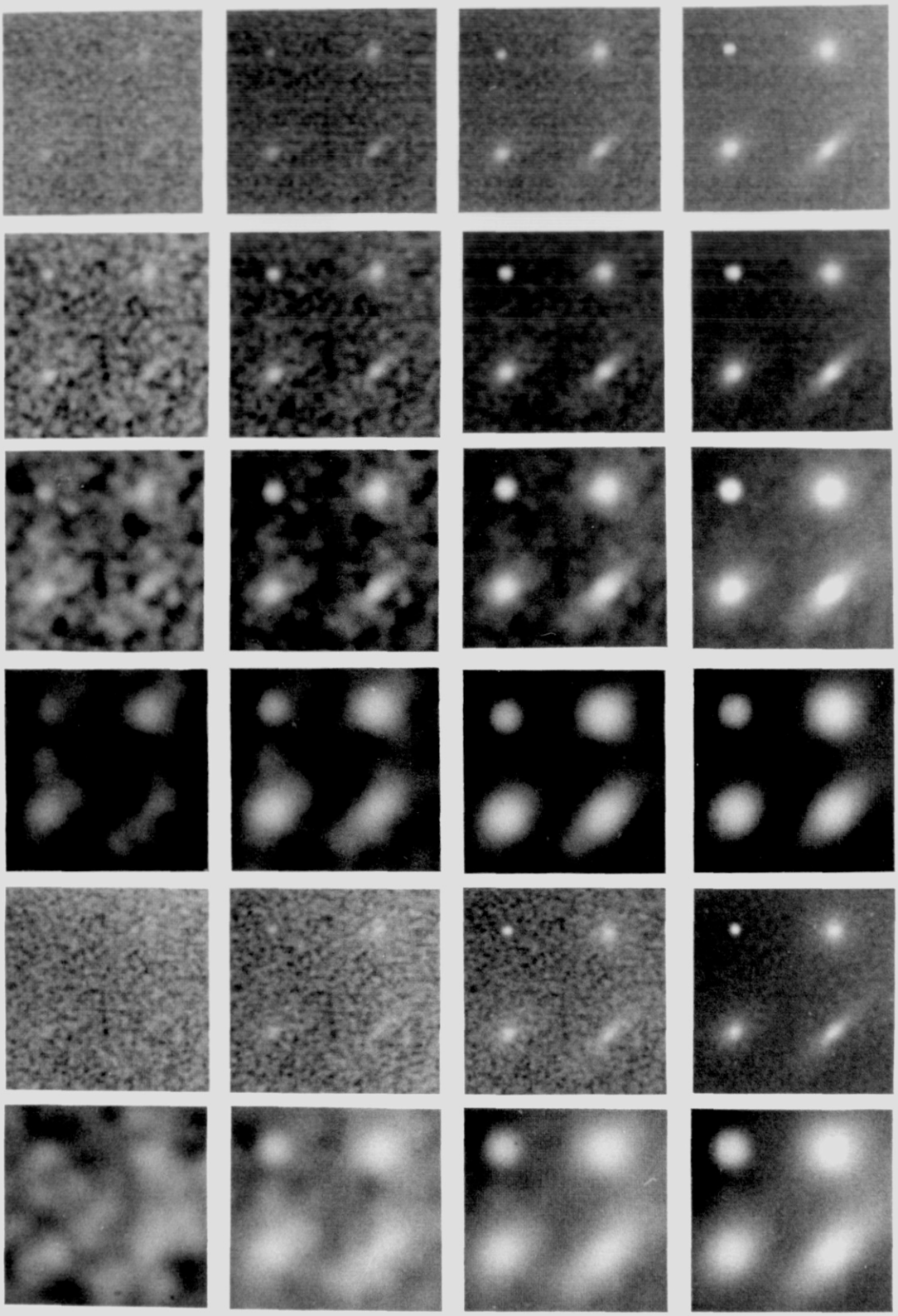
Pl.3.4a(ii) Appearance of double stars (cont).
Ratio 3. Separation reducing by factors of fourth root of 2 (1.19) from 4Ws. Weak component matched filter Sn 12.5.
Upper: filter none (Ws 2 pixels); gaussian Wf/Ws = ½, 1.
Lower: filter none (at Sn 31); square, side 1.2Ws, 2.5Ws.



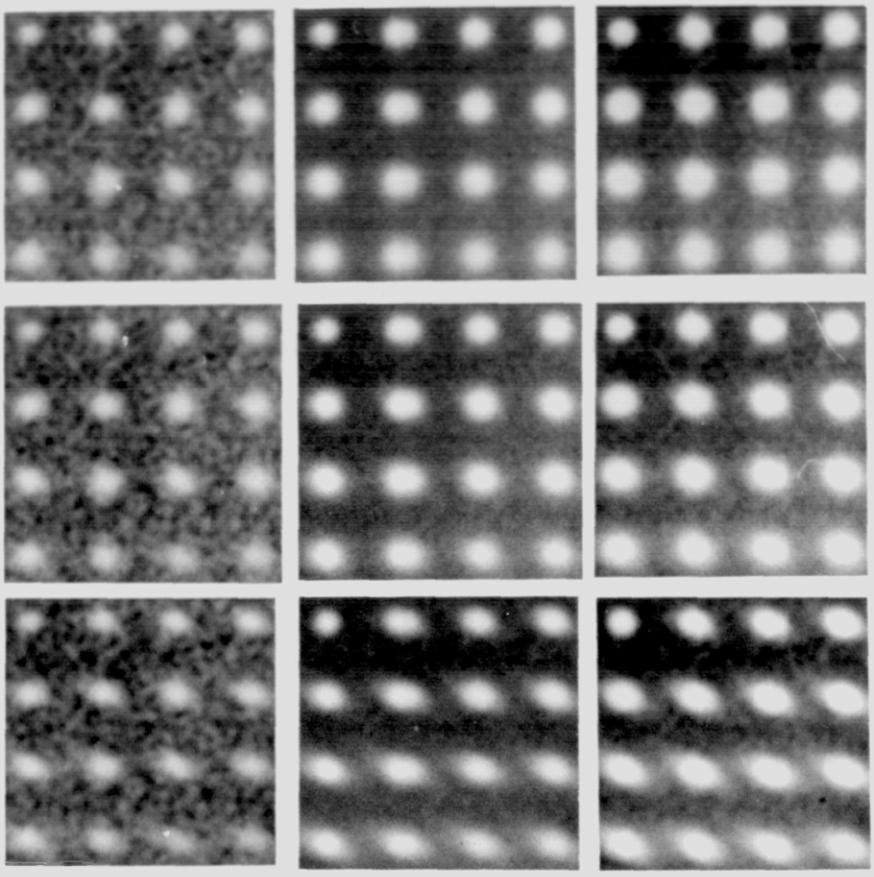
Pl.3.5a Example of deconvolution in one dimension.
Double gaussian line, separation 1.64Wl. Matched filter Sn for weak component 280.
Left to right: single line; double line with noise; after gaussian filtering Wf/Wl = ½; after deconvolution (low overshoot) sharpening factor 1.2, 1.4, 1.7.



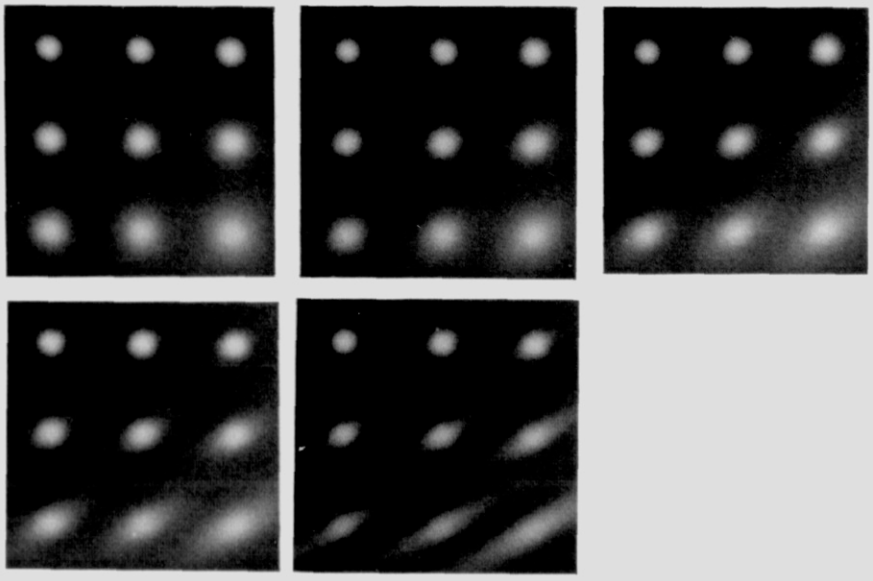
Pl.3.6a Example of deconvolution in two dimensions.
Left to right: low overshoot filter sharpening factor 1.4 (only central part 17 pixels square used); star before and after sharpening; double star separation 1.41Ws before and after sharpening.



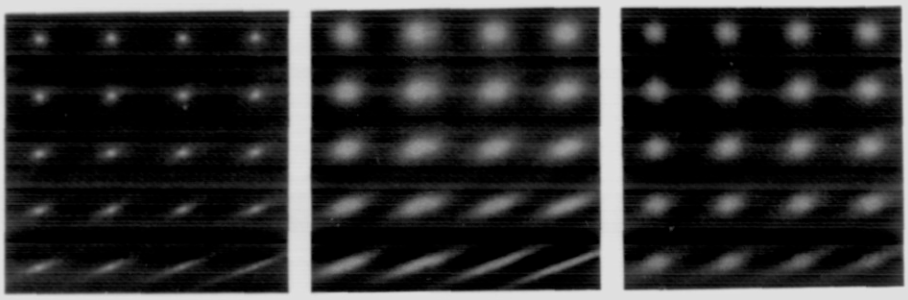
Pl.3.7a(i) Elliptical galaxies: large Re.
Each frame has a star and galaxies of ellipticity 0.0, 0.3, and 0.7, of approximately equal peak brightness when filtered. Brightness increases (l to r) in steps of 1 magnitude, starting at close to the detection threshold.
Re/Ws (t to b): 10, 10, 10, 10, 40, 40.
Filter gaussian, Wf/Ws (t to b): ½, 1, 2, 4, ½, 4.
The right hand frame on line above bottom is 1½ magnitudes brighter than the one to the left.



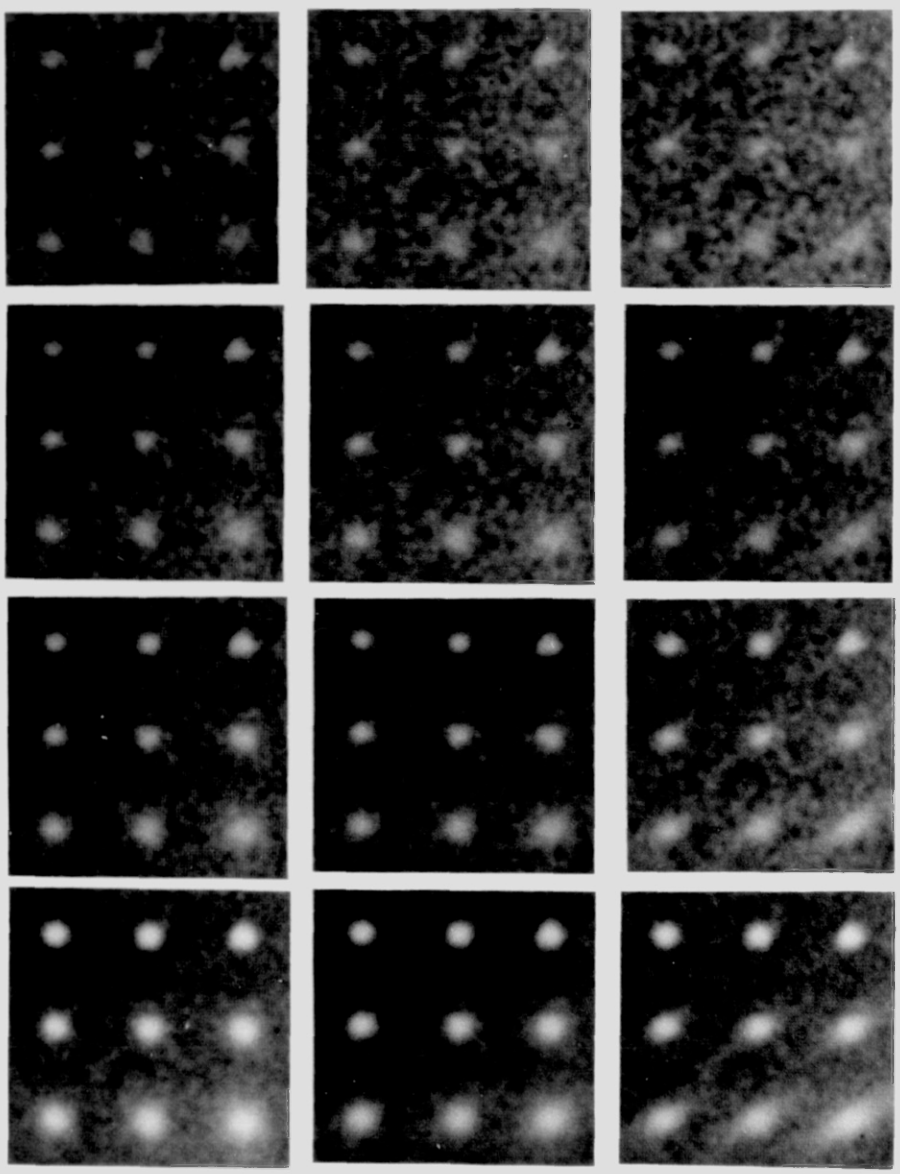
Pl.3.7a(ii) Elliptical galaxies: small Re.
Gaussian filter Wf/Ws=½.
Actual Sn approximately (l to r): 20, 130, 130 (plot heavily saturated).
Ellipticity (t to b): 0.0, 0.3, 0.7.
Re/Ws: 0.44 to 5.0 by factors of 1.41. The first object is a star.



Pl.3.7b(i) Elliptical galaxies: Re.
Convolved with a gaussian Ws=0.022 (relative to width of plot).
Re/Ws: 0.07 to 18 by factors of 2.
Ellipticity (l to r): upper 0.0, 0.3, 0.5; lower 0.7, 0.9.



Ellipticity 0.0 to 0.95 by 0.05, Re 0.2 (relative to width of plot).
Left to right: unfiltered, unfiltered and plot heavily saturated, filtered gaussian Ws/Re=0.1.



Convolved with a gaussian Ws=0.02 (relative to width of plot), filtered gaussian Wf=0.01.
Re/Ws: 0.08 to 20 by factors of 2.
Ellipticity (l to r): 0.0, 0.3, 0.7.
Actual Sn (t to b): 8, 12, 20, 50.

Pl.3.7b(ii) Elliptical galaxies.

SELECT BIBLIOGRAPHY

This bibliography covers a number of the fields touched on in this study. In many cases it has been necessary to be extremely selective, and only important references are given, with emphasis on recent and review papers. And in others the citations are a representative sample. Generally papers published as abstracts only have been omitted, and those in foreign languages unless of exceptional interest.

IMAGE PROCESSING

Image processing has become a large and well established field, which in its non-astronomical aspects is well served by textbooks (1,2), and for some hardware aspects (3). Astronomical image processing in general has been the subject of a series of conferences (4-6, see also 7-13: and for a review 14); some of the individual papers are cited below. Much early work was done at JPL from 1964 (15,16), which gradually developed into astronomical uses (17), and the astronomical literature becomes prolific from the mid 1970s.

1) W.K.Pratt: Digital Image Processing. 1978. Wiley, New York
2) A.Rosenberg, A.C.Kak: Digital Image Processing. 1976. Academic Press, New York
3) J.D.Foley, A.van Dam: Fundamentals of Interactive Computer Graphics. 1982. Addison-Wesley, Reading, Massachusetts
4) Image Processing Techniques in Astronomy. Conference at Utrecht, 1975. C.de Jager, H.Nieuwenhuijzen (eds). D.Reidel, Dordrecht, 1975
5) Image Processing in Astronomy. Proceedings of 5th colloquium on astrophysics, Trieste, 1979. G.Sedmak, M.Capaccioli, R.J.Allen (eds). Osservatorio Astronomico di Trieste, 1979
6) Proceedings of the Conference on Applications of Digital Image Processing to Astronomy. Pasadena, 1980. Proc. SPIE, Vol.264 (1980)

7) Automation in Optical Astronomy. 11th colloquium of IAU, Edinburgh, 1970. H.Seddon, M.J.Smyth (eds). Publ.Royal Obs.Edinburgh, Vol.8 (1971)
8) Meeting on Methodologies for Information Processing. Trieste, 1977 and Rome, 1978. Mem.Soc.Astron.Ital., Vol.49, No.4 (1978)
9) 2nd meeting on Methodologies for Information Processing. Trieste, 1979. G.A.de Biase, G.Sedmak (eds). Mem.Soc. Astron.Ital., Vol.50, No.3 (1979)
10) 3rd meeting on Methodologies for Information Processing. Rome, 1980. G.A.de Biase, G.Sedmak (eds). Mem.Soc.Astron. Ital., Vol.51 (1980)
11) Image Restoration at Optical, Infrared and Radio Wavelengths: Astrophysical Applications. Conference at Bordeaux, 1979. J.Opt.(France), Vol.10, No.6 (1979)
12) Automated Data Retrieval in Astronomy. 64th colloquium of IAU, Strasbourg, 1981. C.Jaschek, W.D.Heintz (eds). D.Reidel, Dordrecht, 1982
13) Astronomy with Schmidt-type Telescopes. 78th colloquium of IAU, Asiago, 1983. M.Capaccioli (ed). D.Reidel, Dordrecht, 1984
14) R.N.Bracewell. 1979. Annu.Rev.Astron.Astrophys., Vol.17, 113-134
15) F.C.Billingsley. 1970. Applied Optics, Vol.9, 289-299
16) D.A.O'Handley, W.B.Green. 1972. Proc.IEEE, Vol.60, 821-828
17) J.J.Lorre, D.J.Lynn, W.D.Benton. 1976. Proc.SPIE, Vol.74, 234-238

The astronomical use of image processing has been closely associcated with the development of plate scanners (18), though it has also been applied to radio maps (19,20), the output of CCD cameras (21), and the like. Automated microdensitometers serve for the scanning of small areas, with the Perkin-Elmer PDS becoming standard (for example 22,23), while a number of special machines, in the UK starting with GALAXY (24-29), and more recently COSMOS (31-35) and APM (37-40), have been built for processing large plates, especially from Schmidts. For comparisons, and other machines, (41-43).

18) Proceedings of Workshop on Astronomical Measuring Machines. Edinburgh, 1982. R.S.Stobie, B.McInnes (eds). Occas.Rep.R. Obs.Edinb., No.10 (1983)
19) R.J.Allen. 1979. in (5), 95-99
20) L.R.D'Addario. 1984. Proc.SPIE, Vol.231, 2-9
21) T.E.McGuire. 1983. Publ.Astron.Soc.Pacific, Vol.95, No.573, 919-924
22) H.Lindgren, G.Lynga, P.Linde, H.O.Hagerbo. 1979. in (5), 65-76
23) W.C.Seitter. 1984. in (13), 159-163
24) P.B.Fellgett. 1970. Optics Technol., Vol.2, No.2, 61-64
25) V.C.Reddish. 1970. Phys.Bull.(GB), Vol.21, 351-352
26) R.H.Stoy. 1970. in (30), 48-53
27) G.S.Walker. 1971. in (7), 103-108

28) N.M.Pratt. 1971. in (7), 109-118
29) C.A.Murray, W.Nicholson. 1975. in (4), 171-184
30) Proper Motions. 7th colloquium of IAU, Minneapolis, 1970. W.J.Luyten (ed). University of Minnesota, 1970
31) N.M.Pratt. 1974. in (36), 299-300
32) N.M.Pratt, R.Martin, L.W.G.Alxeander, G.S.Walker, P.R.Williams. 1975. in (4), 217-222
33) N.M.Pratt. 1977. Vistas Astron., Vol.21, Part 1, 1-42
34) R.S.Stobie, G.M.Smith, R.K.Lutz, R.Martin. 1979. in (5), 48-64
35) R.J.Dodd, I.Dalton. 1980. Astrophys.Space Sci., Vol.69, 319-324
36) New Problems in Astrometry. IAU symposium No.61, Perth, 1973. W.Gliese, C.A.Murray, R.H.Tucker (eds). D.Reidel, Dordrecht, 1974
37) E.J.Kibblewhite. 1971. in (7), 122-123
38) E.J.Kibblewhite, M.T.Bridgeland, T.Hooley, D.Horne. 1975. in (4), 245-246
39) The Automated Photographic Measuring Facility at Cambridge. 1983. Institute of Astronomy, Cambridge
40) E.J.Kibblewhite, M.Bridgeland, P.Bunclark, M.Cawson, M.Irwin. 1984. in (13), 89-97
41) W.J.Luyten. 1971. Proc.National Acad.Sci.USA, Vol.68, 513-516
42) R.M.West. 1975. in (4), 223-228
43) R.Fong, J.G.Godwin, M.R.Green, T.Shanks. 1983. in (18), 61-73

Complete systems for image processing and analysis, usually including plate scanners, have been established to serve most major astronomical centres, for example (44-55). Occcasionally these have been networks serving a number of institutions, such as STARLINK (56-58). While these have often been based on 'super-mini' computers such as the VAX, smaller, less powerful systems have also been described, based on minis (59). The software for some of these has been written in FORTH (60,61). Microcomputers have not been described for serious image processing, having only quite recently become powerful enough to make them attractive. There have also been systems described for processing one dimensional, spectral, data; both self-contained systems on their own hardware, and software packages only (62-70).

44) F.Middleberg, P.Crane. 1979. in (5), 25-36
45) K.Banse, P.Crane, P.Grosbol, F.Middleberg, C.Ounnas, D.Ponz, H.Waldthausen. 1983. Messenger, No.31, 26-28
46) P.Crane. 1984. in (13), 99-106
47) D.C.Wells. 1979. in (5), 1-24
48) R.Albrecht. 1979. in (5), 155-172
49) H.Maehara. 1981. AAS Photo-Bull., No.26, 14-16
50) H.Maehara, T.Yamagata. 1984. in (13), 169-171
51) T.Kreidl, M.Buchholz, C.Winkler. 1979. Messenger, No.19, 21-25
52) F.Macchetto. 1982. in (12), 127-129

53) P.Crane, K.Banse. 1982. Mem.Soc.Astron.Ital., Vol.53, No.1, 19-29
54) D.D.Walker, M.L.Kendall. 1980. J.British Interplanet.Soc., Vol.33, 347-351
55) G.Buscema, G.Iannicola, D.Nanni, G.Pittella, D.Trevese, A.Vignato. 1980. Mem.Soc.Astron.Ital., Vol.51, 451-461
56) D.J.Carnochan. 1982. in (12), 63-65
57) K.P.Tritton. 1982. Mem.Soc.Astron.Ital., Vol.53, No.1, 55-59
58) G.Sedmak. 1982. in (12), 67-71
59) L.Benacchio, M.Capaccioli, G.A.de Biase, P.Santin, G.Sedmak. 1979. in (5), 196-210
60) F.Schiffer, C.A.Harvel, T.R.Gull. 1977. Proc.Southwest Reg. Conf.for Astron.and Astrophys., Vol.3. Texas Tech University. 99-115
61) A.C.Davenhall, P.S.Bunclark, C.W.Fraser, B.J.McLean, J.R.Stapleton, G.C.Stewart. 1981. J.British Interplanet. Soc., Vol.34, 145-151
62) D.W.Peat, A.C.Pemberton. 1970. Observatory, Vol.90, 141-146
63) L.Crivellari, P.Santin, G.Sedmak, L.Rusconi. 1978. Mem.Soc. Astron.Ital., Vol.49, 717-721
64) L.Rusconi, G.Sedmak. 1979. Mem.Soc.Astron.Ital., Vol.50, 429-436
65) F.Pasian, L.Rusconi, G.Sedmak. 1980. Mem.Soc.Astron.Ital., Vol.51, 405-421
66) C.Bonoli, F.Bortoletto, R.Falomo. 1979. Mem.Soc.Astron.Ital., Vol.50, 411-428
67) J.Straeda. 1980. Anglo-Australian Obs.Prepr., No.135, 8pp
68) R.N.Clark. 1980. Publ.Astron.Soc.Pacific, Vol.92, 221-224
69) J.B.Rice. 1981. Publ.Dominion Astrophys.Obs., Vol.16, 1-10
70) L.Rossi, A.Altamore, A.Giangrande, O.Ricciardi, R.Viotti. 1982. Mem.Soc.Astron.Ital., Vol. 53, No.1, 241-247

While digital image processing has become predominant, optical processing is also possible. Coherent optical processing (71) has been applied to astronomy mostly by the Russians, and seems unlikely to ever become a normal technique (72-77). A certain amount of image processing is possible by purely optical means, and this is routinely used, being well suited to enhancing the image quality of large plates, for which digital image processing is not yet practicable (78-81).

71) K.Preston: Coherent Optical Computers. 1972. McGraw Hill, New York
72) S.G.Kuz'menkov, V.G.Chenyj. 1979. Astrometriya i Astrofizika, Vyp.37, 80-84
73) V.N.Dudinov, V.S.Tsvetkova, V.A.Krishtal'. 1979. in New Techniques in Astronomy. N.N.Mikhel'son (ed). Nauka, Leningrad. 60-71
74) B.Bates, D.L.Giaretta, P.J.P.Sweeney. 1980. Astron. Astrophys., Vol.90, 318-323
75) R.A.Sarkisyan. 1980. Soobshch.Byurakan Obs., Vyp.52, 109-117
76) A.O.Bakrunov, O.S.Sladkov, M.F.Shadbanov, I.V.Shchukin. 1981. Astron.Tsirk., No.1195, 1-3

77) V.S.Tsvetkova, V.G.Chernyj. 1984. Soviet Astron.Lett., Vol.10
78) D.F.Malin. 1981. Anglo-Australian Obs.Prepr., No.148, 14pp
79) D.F.Malin. 1981. J.Photogr.Sci., Vol.29, 199-205
80) D.F.Malin. 1984. in (13), 57-72
81) B.W.Hadley. 1984. in (13), 73-76

The properties of the photographic image and the theory of its noise are given in (82). Modern astronomical photographic methods are documented in (83-85).

82) J.C.Dainty, R.Shaw: Image Science. 1974. Academic Press, London
83) D.W.Latham. 1974. in Methods of Experimental Physics, Vol.12A. Academic Press, London. 221-235
84) A.G.Smith, A.A.Hoag. 1979. Annu.Rev.Astron.Astrophys., Vol.17, 43-72
85) Modern Techniques in Astronomical Photography. Proceedings of ESO workshop, Geneva, 1978. R.M.West, J.L.Heudier (eds). ESO, Garching bei Munchen, 1978

RESOLUTION

Resolution from ground based telescopes, and atmospheric seeing, is discussed in (86-90). For the observed profiles of star images (92-96). For typical examples of measurements of seeing (97-101). Diffraction in the telescope is a less significant cause of degradation than the atmosphere, and it can be modified by apodization (102-104); for a practical use, to supress 'spider' diffraction see (105,106). For an example of the effect of mirror imperfections (107).

86) N.J.Woolf. 1982. Annu.Rev.Astron.Astrophys., Vol.20, 367-398
87) A.T.Young. 1974. Astrophys. Journ., Vol.189, 587-604
88) F.Roddier. 1981. Progress in Optics, Vol.19, 283-376
89) D.S.Brown. 1982. in (91), 85-92
90) Site Testing for Future Large Telescopes. Proceedings of ESO workshop, La Silla, 1983. A.Ardeberg, L.Woltjer (eds). ESO, Garching bei Munchen, 1984
91) Instrumentation for Astronomy with Large Optical Telescopes. IAU colloquium No.67, Zelenchukskaya, 1981. C.M.Humphries (ed). D.Reidel, Dordrecht, 1982
92) I.R.King. 1971. Publ.Astron.Soc.Pacific, Vol.83, 199-201
93) J.Piccirillo. 1973. Publ.Astron.Soc.Pacific, Vol.85, 278-280
94) A.I.Khaliullina, Kh.F.Khaliullin, A.M.Cherepashchuk. 1979. Soviet Astron., Vol.23, No.4
95) G.Lynga, P.Linde. 1983. in (18), 105-110
96) A.C.Edwards. 1984. Mon.Not.R.Astron.Soc., Vol.207, No.3, 517-531
97) M.F.Walker. 1970. Publ.Astron.Soc.Pacific, Vol.82, 672-698
98) M.F.Walker. 1971. Publ.Astron.Soc.Pacific, Vol.83, 401-422
99) B.McInnes, M.F.Walker. 1974. Publ.Astron.Soc.Pacific, Vol.86, 529-544

100) H.A.Abt. 1980. Publ.Astron.Soc.Pacific, Vol.92, 103-107
101) H.M.Dyck, R.R.Howell. 1982. Publ.Astron.Soc.Pacific, Vol.95, No.572, 786-791
102) A.M.Clements, J.E.Wilkins. 1974. J.Opt.Soc.America, Vol.64, 23-27
103) A.V.Lenskij. 1982. Soviet Astron., Vol.26, No.3
104) A.B.Schultz, T.V.Frazier, E.Kosso. 1984. Appl.Opt., Vol.23, No.12, 1914-1916
105) A.Couder. 1934. L'Astronomie, January. Trans. in Amateur Telescope Making, Book 2. A.G.Ingalls (ed), Scientific American Inc., 1937
106) J.L.Richter. 1984. Appl.Opt., Vol.23, No.12, 1907-1913
107) F.Vakili. 1979. C.R.Acad.Sci.Paris, Tome 288, Ser.B, 61-63

Filtering to enhance images, reducing the effect of noise, or improving resolution, has been extensively discussed, and astronomical examples given (108-117).

The maximum entropy method, which is an alternative to linear filtering as a way of enhancing resolution, originated (118,119) and is well described (120). See also (121,122), for problems and limitations (123), and astronomical examples (124,125). It has been much used in radio (126-128) and X-ray (129,130) astronomy, because of its ability to deal with absent data points and extreme photon noise. The CLEAN algorithm used in radio astronomy has very similar properties (131-134).

Other methods of deconvolution have occasionally been used for one dimensional problems, such as removing the effect of seeing from galaxy brightness profiles (135-139).

108) P.H.Richter. 1974. in Instrumentation in Astronomy II. Proceedings of conference in Tucson, 1974. SPIE, 215-221
109) H.Arp, J.J.Lorre. 1976. Astrophys.J., Vol.210, 58-64
110) F.Paisan, G.Sedmak, L.Rusconi. 1978. Mem.Soc.Astron.Ital., Vol.49, 745-751
111) F.Paisan, G.Sedmak. 1978. Mem.Soc.Astron.Ital., Vol.49, 753-758
112) M.L.Malagnini, G.L.Sicuranza. 1978. Astron.Astrophys., Vol.70, 579-582
113) A.Andronico, F.Bernardi, P.L.Casalini, G.Marchetti. 1979. Mem.Soc.Astron.Ital., Vol.50, 483-495
114) V.Cappellini. 1979. in (5), 258-267
115) M.R.S.Hawkins. 1979. Astron.Astrophys., Vol.76, 46-49
116) A.A.Korovyakovskaya, Yu.P.Korovyakovskij. 1982. Soviet Astron., Vol.26, No.1
117) P.A.Wehinger, S.Wyckoff, T.Gehren, H.Spinrad. 1983. in Early Evolution of the Universe and its Present Structure. IAU symposium No.104. D.Reidel, Dordrecht.
118) J.P.Burg. 1967. Proc.37th Meeting Soc.Explor.Geophys.
119) J.G.Abels. 1974. Astron.Astrophys.Suppl., Vol.15, 383
120) S.F.Gull, G.J.Daniell. 1978. Nature, Vol.272, 686-690
121) B.R.Frieden, D.C.Wells. 1978. J.Opt.Soc.America, Vol.68, 93-103

122) R.Nityananda, R.Narayan. 1982. J.Astrophys.Astron., Vol.3, No.4, 419-450
123) M.M.Komesaroff, R.Narayan, R.Nityananda. 1981. Astron. Astrophys., Vol.93, 269-281
124) R.K.Bryan, J.Skilling. 1980. Mon.Not.R.Astron.Soc., Vol.191, 69-79
125) J.J.Lorre, J.-L.Nieto. 1984. Astron.Astrophys., Vol.130, No.1, 167-170
126) S.J.Wernecke. 1977. Radio Sci., Vol.12, 831-844
127) M.Sanroma, R.Estalella. 1983. in Very Long Baseline Interferometry Techniques. Proceedings of conference in Toulouse. Cepadues-Editiones, Toulouse. 391-407
128) M.Sanroma, R.Estalella. 1984. Astron.Astrophys., Vol.133, No.2, 299-306
129) J.Skilling, A.W.Strong, K.Bennett. 1979. Mon.Not.R.Astron. Soc., Vol.187, 145-152
130) R.Willingale. 1981. Mon.Not.R.Astron.Soc., Vol.194, 359-364
131) J.A.Hogbom. 1974. Astron.Astrophys.Suppl., Vol.15, 417-426
132) A.Segalovitz, B.R.Frieden. 1978. Astron.Astrophys., Vol.70, 335-343
133) U.J.Schwarz. 1978. Astron.Astrophys., Vol.65, 345-356
134) U.J.Schwarz. 1979. in Image Formation from Coherence Functions in Astronomy. IAU colloquium No.49. D.Reidel, Dordrecht. 261-275
135) K.Taylor. 1974. Astrophys.Space Sci., Vol.26, 327-336
136) A.E.Clements, B.M.Herman. 1974. Applied Optics, Vol.13, 2947-2956
137) O.Bendinelli, G.Parmeggiani, F.Zavatti. 1981. in (140), 115-122
138) O.Bendinelli, G.Parmeggiani, F.Zavatti. 1982. Astrophys. Space Sci., Vol.83, 239-246
139) F.Zavatti, O.Bendinelli, G.Parmeggiani. 1984. Astron. Astrophys., Vol.134, No.1, 99-104

The alternative is to obtain better resolution observationally. For some general discussions (140-145). The effect of site, and fluctuations of seeing, have been referenced above. Examples of exceptional resolution at long and short exposures are given in (147 and 148). For improved resolution by selecting from a sequence of short exposure pictures (149). The capabilities of visual observation are exemplified by double star measurements (150-152): typical work is given (153-155). For the balloon borne Stratoscope II telescope (156-158); for the Space Telescope (159,160).

140) Scientific Importance of High Angular Resolution at Optical and Infrared Wavelengths. ESO Conference, Garching, 1981. M.H.Ulrich, K.Kjar (eds). ESO, Garching bei Munchen, 1981
141) J.Rosch. 1972. Space Research XII, Vol.2, 1633-1656
142) M.J.Aucremanne. 1972. Space Research XII, Vol.2, 1787-1807
143) P.Nisenson, R.Stachnik, C.Papaliolios. 1980. in (146), 401-407
144) F.Roddier. 1981. in (140), 5-23

145) R.N.Wilson. 1981. in (140), 67-83
146) Optical and Infrared Telescopes for the 1990s. KPNO conference, Tucson, 1980. A.Hewitt (ed). Kitt Peak National Observatory, 2 vol., 1980
147) J.Andersen, B.Niss. 1979. Messenger, No.17, 14-15
148) L.D.Parfinenko, V.F.Mikhalev. 1978. Soln.Dannye 1978 Byull., No.8, 92-97
149) B.Fort, J.P.Picat, M.Cailloux, N.Mauron, M.Dreux, T.Fauconnier. 1984. Astron.Astrophys., Vol.135, No.2, 356-360
150) P.Couteau: Observing Visual Double Stars.(trans. A.H.Batten). 1981. MIT Press
151) A.A.Takovinin. 1983. Soviet Astron.Lett., Vol.9, No.5, 327-329
152) C.E.Worley. 1983. Lowell Obs.Bull., No.167, Vol.9, No.1, 1-9
153) W.Heintz. 1978. Astrophys.J., Suppl.Ser., Vol.37, 343-370
154) P.Couteau. 1979. Astron.Astrophys., Suppl.Ser., Vol.36, 11-24
155) P.J.Morel. 1983. Astron.Astrophys., Suppl.Ser., Vol.54, No.1, 39-46
156) D.J.McCarthy. 1969. IEEE Trans.Aerospace Electronic Systems, Vol.AES-5, No.2, 323-329
157) J.Ashbrook. 1970. Sky and Telescope, Vol.39, 394-396
158) R.E.Danielson, B.D.Savage, M.Schwarzschild, M.Tomasko. 1970. Bull.American Astron.Soc., Vol.2, 307
159) Scientific Research with the Space Telescope. IAU colloquium No.54, Princeton, 1979. M.S.Longair, J.W.Warner (eds). NASA, 1979
160) The Space Telescope Observatory. IAU, Greece, 1981. D.N.B.Hall (ed). NASA, 1982

Radically different observational methods of obtaining high resolution are the interferometric techniques. For general discussions (161-165).

Speckle has become the best established. For origins (166-168), and reviews (169-174). It has become a standard method of double star observation (175-176). Nearly diffraction limited resolution on the largest telescopes has been achieved (177-180), and it has been applied to systems as faint as Pluto/Charon (181,182). For theory and noise properties (183,184).

Intensity interferometry (185) and two telescope Michelson interferometry (186-193, results 194-199) can achieve much higher resolution, the second on much fainter objects. Active optics is potentially capable of achieving similar resolution to speckle but with better sensitivity (200, also 201-210): only very simple systems have been demonstrated on stars (212,213).

161) R.Hanbury Brown. 1983. in Understanding the Universe: the Impact of Space Astronomy. IAU, Vienna, 1982. D.Reidel, Dordrecht
162) H.Van de Stadt. 1975. Space Sci.Rev., Vol.17, 673-676
163) D.V.Korol'kov, O.I.Krat. 1976. Soviet Astron., Vol.20, No.3
164) A.Labeyrie. 1977. Progress in Optics, Vol.XIV, 49-87

165) A.Labeyrie. 1978. Annu.Rev.Astron.Astrophys., Vol.16, 77-102
166) A.Labeyrie. 1970. Astron.Astrophys., Vol.6, 85-87
167) J.C.Dainty. 1973. Optics Commun.(Netherlands), Vol.7, 129-134
168) C.Y.C.Liu, A.W.Lohmann. 1973. Optics Commun.(Netherlands), Vol.8, 372-377
169) R.Jones, C.Wykes: Holographic and Speckle Interferometry. 1983. Cambridge University Press
170) A.Labeyrie. 1981. in (140), 87-93
171) G.Weigelt. 1981. in (140), 95-114
172) J.C.Dainty. 1981. in Recent Advances in Observational Astronomy. Symposium in Ensenada, Mexico, 1979
173) R.H.T.Bates. 1982. Phys.Rep., Vol.90, No.4, 203-297
174) P.Nisenson, J.Apt, R.Goody, C.Papaliolios. 1983. Icarus, Vol.53, No.3, 465-478
175) H.A.McAlister. 1983. Lowell Obs.Bull., No.167, Vol.9, No.1, 125-143
176) G.Weigelt. 1983. Lowell Obs.Bull., No.167, Vol.9, No.1, 144-153
177) A.Labeyrie. 1974. Nouvelle Rev.Optique (France), Vol.5, 141-151
178) H.A.McAlister. 1979. Astrophys.J., Vol.230, 497-501
179) Yu.Yu.Balega, A.Blazit, D.Bonneau, L.Koechlin, R.Foy, A.Labeyrie. 1982. Astron.Astrophys., Vol.115, 253-256
180) Yu.Yu.Balega, V.P.Ryadchnko. 1984. Soviet Astron.Lett., Vol.10
181) D.Bonneau, R.Foy. 1980. Astron.Astrophys., Vol.92, L1-L4
182) G.Baier, G.Weigelt. 1984. Angewandte Optik, 36-37
183) J.C.Dainty. 1974. Mon.Not.R.Astron.Soc., Vol.169, 631-641
184) M.G.Miller. 1977. J.Opt.Soc.America, Vol.67, 1176-1184
185) R.Hanbury Brown: The Intensity Interferometer. 1974. Taylor and Francis, London
186) A.H.Greenaway, J.C.Dainty. 1978. Opt.Acta, Vol.25, No.3, 181-189
187) M.Shao. 1979. in Modern Astrometry. IAU colloquium No.48, Vienna, 1978. Institute of Astronomy, Vienna
188) A.H.Greenaway. 1979. Opt.Acta, Vol.26, 1147-1171
189) A.H.Greenaway. 1980. in (146), 755-785
190) W.Walker. 1980. J.Opt.Soc.America, Vol.70, 1096-1100
191) W.J.Tango, R.Q.Twiss. 1980. Progress in Optics, Vol.XVII, 241-277
192) J.Davis. 1983. Lowell Obs.Bull., No.167, Vol.9, No.1, 191-201
193) R.Hanbury Brown. 1984. J.Astrophys.Astron., Vol.5, No.1, 19-30
194) A.Labeyrie. 1975. Astrophys.J.(Lett.), Vol.196, L71-L75
195) A.Blazit, D.Bonneau, L.Koechlin. 1977. C.R.Acad.Sci.Paris, Tome 285, Ser.B, 149-152
196) M.Shao, D.H.Staelin. 1980. Appl.Opt., Vol.19, 1519-1522
197) F.Vakili, D.Bonneau, L.Koecklin. 1981. in (140), 399-404
198) D.Bonneau, L.Koechlin, J.L.Oneto, F.Vakili. 1981. Astron. Astrophys., Vol.103, 28-34

199) L.Koechlin, F.Vakili, M.Faucherre, P.Di Benedetto, J.C.Conti. 1983. in Very Long Baseline Interferometry Techniques. Conference in Toulouse. Cepadues-Editiones, Toulouse. 311-317
200) J.W.Hardy. 1978. Proc.IEEE, Vol.66, 651-697
201) R.A.Muller, A.Buffington. 1974. J.Opt.Soc.America, Vol.64, 1200-1210
202) R.H.Dicke. 1975. Astrophys.J., Vol.198, 605-615
203) F.J.Dyson. 1975. J.Opt.Soc.America, Vol.65, 551-558
204) J.W.Hardy, J.E.Lefebvre, C.L.Koliopoulos. 1977. J.Opt.Soc. America, Vol.64, 360-369
205) F.J.Dyson. 1978. in (211), 439-444
206) S.L.McCall, A.Passner. 1978. in Adaptive Optics and Short Wavelength Sources. Addison-Wesley, London
207) G.L.Bourdet, A.M.Clavron, G.M.Mullot, A.G.Orszag. 1978. in (211), 445-453
208) J.W.Hardy. 1978. in (211), 455-467
209) J.R.P.Angel. 1981. in (140), 53-60
210) J.W.Hardy. 1981. in (140), 25-39
211) Optical Telescopes of the Future. ESO conference, Geneva, 1977. F.Pacini, W.Richter, R.N.Wilson (eds). ESO, Garching bei Munchen, 1978
212) S.L.McCall, T.R.Brown, A.Passner. 1977. Astrophys.J., Vol.211, 463-468
213) A.Buffington, F.S.Crawford, S.M.Pollaine, C.D.Orth, R.A.Muller. 1978. Science, Vol.200, 489-494

GENERAL

Photometry (214-223) and astrometry (224-227) with plate scanners are well documented. For methods of analysis (228-231). For effect of noise (232); completeness of star counts (233); optimum scanning and reduction parameters (234).

214) R.H.Dodd, H.T.MacGillivray, G.Smith, L.A.Ellery. 1979. in (5), 360-370
215) P.B.Stetson. 1979. Astron.J., Vol.84, 1056-1066
216) R.Buonanno, R.Coluzzi, C.E.Corsi, G.A.de Biase, I.Ferraro. 1979. Mem.Soc.Astron.Ital., Vol.50, 451-454
217) R.Albrecht. 1980. in New Techniques in Stellar Photometry and Polarimetry. IAU, Montreal, 1979. Ric.Astron., Vol.10, special issue. 9-17
218) R.Buonanno, G.Buscema, C.Corsi, G.Iannicola. 1980. Mem.Soc. Astron.Ital., Vol.51, 483-489
219) K.Janes. 1982. in (12), 237-239
220) G.Gilmore. 1983. in (18), 259-296
221) S.Phillips. 1983. in (18), 173-184
222) N.Reid. 1983. in (18), 1-14
223) G.Gilmore. 1984. in (13), 77-87
224) R.J.Dodd. 1972. Astron.J., Vol.77, 306-311
225) L.-T.G.Chiu. 1977. Astron.J., Vol.82, 842-848
226) R.F.Green, M.E.Morrill. 1978. Publ.Astron.Soc.Pacific, Vol.90, 601-606

227) L.-T.G.Chiu, W.F.van Altena, P.B.Stetson. 1979. in (5), 240-251
228) W.F.van Altena, L.H.Auer. 1975. in (4), 411-418
229) L.H.Auer, W.F.van Altena. 1978. Astron.J., Vol.83, 531-537
230) L.H.Auer, W.F.van Altena. 1978. Astron.J., Vol.83, 1468
231) M.J.Irwin, P.Hall. 1983. in (18), 111-121
232) J.-F.Lee, W.F.van Altena. 1983. Astron.J., Vol.88, No.11, 1683-1689
233) G.M.Richter. 1980. Astron.Nachr., Band 301, 157-163
234) S.Okamura, A.C.Davenhall, H.T.MacGillivray. 1983. in (18), 95-104

Discrimination between star and galaxy images has been extensively studied as a necessary aspect of automatic plate analysis and object counts (for example 235-239). Techniques are described in (241-251). Discrimination methods have also been applied to double star images (252,253) and to classifying galaxy types (254,255). For tests of automatic galaxy counts and classifications (256,257).

235) J.A.Tyson, J.F.Jarvis. 1979. Astrophys.J., Lett., Vol.230, L153-L156
236) H.T.MacGillivray, R.J.Dodd. 1980. Astrophys.Space Sci., Vol.72, 315-318
237) H.T.MacGillivray, R.J.Dodd. 1983. in (240), 157-167
238) S.Phillips. 1983. in (240), 169-172
239) H.T.MacGillivray, R.J.Dodd. 1984. in (13), 125-128
240) Clustering in the Universe. Proceedings of a colloquium, Meudon, 1982. D.Gerbal, A.Mazure (eds). Editiones Frontieres, Gif sur Yvette, 1983
241) H.T.MacGillivray, R.Martin, N.M.Pratt, V.C.Reddish, H.Seddon, L.W.G.Alexander, G.S.Walker, P.R.Williams. 1976. Mon.Not.R.Astron.Soc., Vol.176, 265-274
242) W.Sebok. 1979. Astron.J., Vol.84, 1526-1536
243) J.A.Tyson, J.F.Jarvis. 1980. in Objects of High Redshift. IAU symposium No.92. D.Reidel, Dordrecht. 1-8
244) R.G.Kron. 1980. Astrophys.J., Suppl.Ser., Vol.43, 305-325
245) J.F.Jarvis, J.A.Tyson. 1981. Astron.J., Vol.86, 476-495
246) S.Bianchi, G.Gavazzi, P.Mussio. 1982. Mem.Soc.Astron.Ital., Vol.53, No.1, 249-265
247) M.L.Malagnini, G.L.Sicuanza. 1982. Mem.Soc.Astron.Ital., Vol.53, 267-277
248) M.L.Malagnini. 1983. ESA Spec.Publ., ESA SP-201, 69-72
249) J.A.Tyson. 1984. in (13), 489-498
250) T.Shanks, P.R.F.Stevenson, R.Fong, H.T.MacGillivray. 1984. in (13), 499-505
251) M.L.Malagnini, M.Pucillo, P.Santin, G.Sedmak,G.L.Sciuranza. 1984. in (13), 133-135
252) W.G.Bagnuolo. 1982. Mon.Not.R.Astron.Soc., Vol.200, 1113-1122
253) A.Bijaoui. 1983. in (18), 41-42
254) G.Sedmak, M.L.Trujillo Lamas. 1981. Astron.Astrophys., Vol.104, 93-98

255) M.Thonnat. 1983. Bull.Inf.Cent.Donnees Stellaires, No.25, 17-18
256) H.T.MacGillivray, R.J.Dodd. 1983. in (18), 195-204
257) A.Wirth. 1984. in (13), 129-131

The effects of seeing on the shapes and profiles of elliptical galaxies have been studied (258-262), and the dependence of apparent profile on ellipticity (263). For various selection effects and classification errors (264-269).

258) F.Schweizer. 1979. Astrophys.J., Vol.223, 23-24
259) M.Capaccioli, R.Rampazzo. 1980. Mem.Soc.Astron.Ital., Vol.51, 491-496
260) F.Schweizer. 1981. Astron.J., Vol.86, 622-666
261) G.F.R.Ellis, L.Lawrence, G.B.Brundrit. 1982. Mon.Notes R. Astron.Soc.S.Afr., Vol.41, Nos.11-12, 107-117
262) B.Thomsen, S.Frandsen. 1983. Astron.J., Vol.88, No.6, 789-803
263) J.-L.Nieto. 1982. Astron.Astrophys., Suppl.Ser., Vol.47, 535-540
264) M.J.Disney. 1976. Nature, Vol.263, 573-575
265) S.van den Bergh. 1977. Observatory, Vol.97, 81-83
266) G.de Vaucouleurs, A.de Vaucouleurs, H.G.Corwin. 1978. Astron.J., Vol.83, 1356-1359
267) R.J.Allen, F.H.Shu. 1979. Astrophys.J., Vol.227, 67-72
268) C.Pritchet, M.I.Kline. 1981. Astron.J., Vol.86, 1859-1869
269) M.J.Disney, S.Phillips. 1983. Mon.Not.R.Astron.Soc., Vol.205, No.3, 1253-1265

Processing spectra is discussed in (270-275), and methods are assessed in (276,277). Reconstructing spectral images is described in (278). For line profiles in the presence of several broadening effects (279), and decomposing such profiles (280,281). Instrumental profiles are discussed in (282,283).

270) L.Rusconi, G.Sedmak. 1971. Astron.Astrophys., Vol.10, 469-473
271) L.Rusconi, G.Sedmak. 1971. in (7), 188-191
272) T.J.Ulrych, J.R.Auman, J.A.Eilek, G.A.H.Walker. 1972. Astron.Astrophys., Vol.21, 125-130
273) J.J.Lorre. 1973. Astron.J., Vol.78, 67-73, 159-161
274) G.A.de Biase, F.Sacchetti, D.Trevese. 1974. J.Opt.Soc. America, Vol.64, 335-338
275) T.Kipper, J.Sitska. 1975. Tartu Astron.Obs., Teated, No.53
276) A.Delcroix. 1980. Bull.Soc.R.Sci.Liege, 49e annee, 244-277
277) B.Bates, B.B.Chaudhuri. 1983. Astron.Astrophys., Vol.123, No.2, 253-256
278) C.Barbieri, R.Mottola. 1979. G.Astron., Vol.5, 227-233
279) J.Humlicek. 1983. J.Quant.Spectrosc.Radiat.Transfer, Vol.29, No.2, 125-130
280) I.Sattarov. 1978. Soviet Astron., Vol.22, No.3

281) R.W.Patch. 1979. J.Quant.Spectrosc.Radiat.Transfer, Vol.22, 273-286
282) D.Dravins. 1975. Rep.Obs.Lund, No.5, 241-256
283) C.F.W.Harmer, D.L.Harmer. 1978. in High Resolution Spectrometry. 4th international colloquium on Astrophysics, Trieste, 1978. Osservatorio Astronomico di Trieste. 627-636

There are a few papers in Russian that are perhaps quite closely related to some aspects of this study. On modelling astronomical images, their image quality and limiting magnitude, (284-286). On correcting spectral line profiles (287). On limiting magnitude and resolving power (288,289).

284) G.G.Lengauer. 1973. Izv.Glav.Astron.Obs.Pulkovo, No.191, Astrofiz.i Astrometr., 56-66
285) G.G.Lengauer, O.A.Melnikov. 1973. Izv.Glav.Astron.Obs. Pulkovo, No.191, Astrofiz.i Astrometr., 67-82
286) G.G.Lengauer. 1974. Izv.Glav.Astron.Obs.Pulkovo, No.192, Astrofiz.i Astrometr., 148-162
287) Sh.M.Sharipov. 1982. Soln.Dannye, Byull., 1982, No.9, 88-94
288) O.A.Melnikov, A.N.Gerashchenko. 1970. Izv.Glav.Astron.Obs. Pulkovo, No.185, 248-267
289) E.V.Morozova. 1970. Izv.Glav.Astron.Obs.Pulkovo, No.185, 268-278