CLM - P 320

UKAEA RESEARCH GROUP

Preprint

APPLICATIONS OF INTERACTIVE COMPUTING IN A SCIENTIFIC ENVIRONMENT

T J MARTIN
A SYKES

CULHAM LABORATORY
Abingdon Berkshire

1972

CLM-P 320

APPLICATIONS OF INTERACTIVE COMPUTING IN A SCIENTIFIC ENVIRONMENT

T. J. Martin and A. Sykes

(Paper to be presented at ONLINE 72 meeting at Brunel University, 4-7 September 1972)

ABSTRACT

Interactive computing has been used at the Culham Laboratory to solve a wide range of problems. We here give examples from the fields of physics, engineering and mathematics, and use these to demonstrate the techniques used and the benefits obtained. In particular, we find that when applied to suitable problems, the interactive approach can enable the scientist to obtain a far better understanding of his problem than could be gained by conventional batch computing.

UKAEA Research Group
Culham Laboratory
Abingdon
Berks.

August 1972

1. Introduction

In this paper we specifically want to consider the suitability (or otherwise) of the interactive approach to given problems arising in a research environment. We consider the basic criterion to be - will a solution process benefit from human intervention, or can it be fully automated? The potential value of man machine interaction is easily demonstrated: consider Fig. 1. It is obvious to

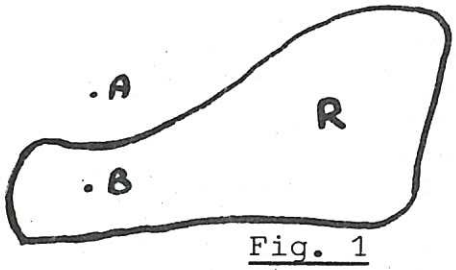

Fig. 1

the human eye that point A is outside the general region R and point B is inside. But to get a computer to recognise this is a non-trivial problem!

We first describe the hardware configurations used at Culham Laboratory, and the available software. Next we discuss several problem areas in detail, noting whether they are amenable to interactive study. We describe suitable programming techniques, and estimate the benefits obtainable from interactive computing.

2. Hardware and software configuration

Most of the work described here was done on an ICL KDF9 computer, coupled to a DEC PDP8 with a DEC 338 display unit, running under the EGDON 3 operating system. The EXECUTE facility of the COTAN multi-access system was used for interactive work (1,2).This facility was, however, only available to privileged users, as the whole 32K KDF9 core had to be rolled-in, rolled-out for interactions.

The present system comprises an ICL 4-70 with 640 K bytes of core storage, coupled to a CTL Modular One, a Cossor CSD 1000 refresh display, and several storage tube displays. The Multijob operating system is used, and this normally provides two roll-in roll out streams, each of 96 Kb (approximately 25,000 32-bit words), some small service streams, and a 200 Kb production stream. At off-peak times, larger streams of up to 500 Kb are available. The system is ideally suited to interactive work.

The GHOST graphics package (3,4) has been developed into a very flexible system; graphical information is usually written to disc files, which the user may process onto teletype, line printer, display, storage tube or (more usually) the CI120 microfilm recorder. Interactive programs short circuit the intermediate storage, processing information directly onto the output device; we discuss the special graphics requirements of interactive programs in Section 4.

3. Applications

a. Field design calculations

We examine here two actual problems in field design. They are chosen to illustrate the suitability question, for one is ideally suited to interactive study; the other is more efficiently done by standard off-line computing.

Example 1: It is required to obtain as uniform a field as possible over the length A B by varying the positions of 40 identical coils, each carrying the same current. Fig. 2(a) shows the field on the axis of a single coil: Fig. 2(b) shows an approximate solution.

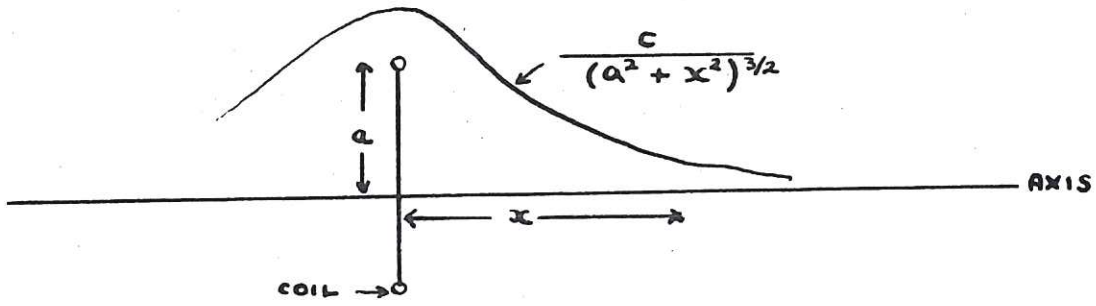

Fig. 2(a)

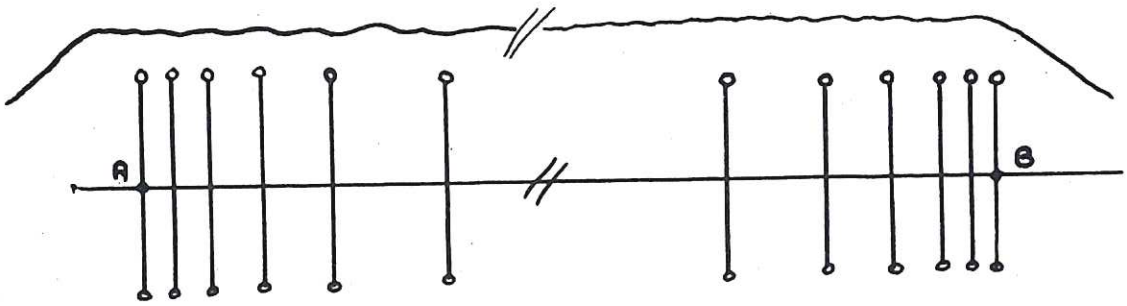

Fig. 2(b)

Although non-linear, the field due to each coil is a well behaved function of its position; the total field at any point of A B is similarly a well-behaved function of the positions of the coils, and the optimum positions may

easily be found by a least squares minimisation technique. This is a routine process, ideally suited to automatic computation; basically, each of the 40 parameters is slightly varied in turn until no further reduction is obtained -- typically after several thousand variations.

Note that to adjust the position of each coil interactively would be a very tedious and time consuming process.

Example 2: we wish to design an electron beam focussing device in a hemispherical annulus, by choosing plate potentials P_1, P_2, P_3 and various other parameters (Fig.3).

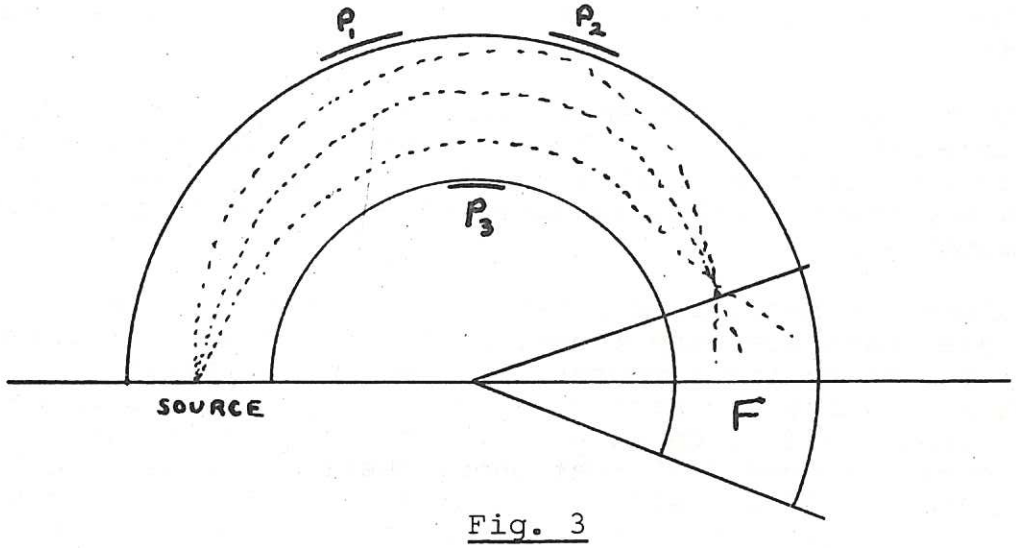

Fig. 3

As in Example 1, we can easily compute the field at any point. However, the trace of each electron orbit in the focussing region F is a very complicated function of the field at each point of its trajectory, and so is a very ill-conditioned and unpredictable function of the design parameters. Hence, even if the problem can be formulated as a least squares minimisation problem (and this is by no means obvious), a global minimum is unlikely to be found by any automatic process.

So we proceed interactively as follows. For each choice of values of the parameters, we compute the orbits for a small range of source emission angles, and show these orbits on the display. The engineer can immediately recognise the effectiveness of the parameter set and, by adjusting each parameter in turn, can in a few minutes develop an understanding of the device that would be difficult to obtain in weeks of batch runs.

To summarise, the Example 1 is a routine problem; the effect of varying any parameter is clearly defined; the problem can be programmed for purely automatic computation.

Example 2 however is a research type problem - we are investigating the unknown response of a complex system; this is best done by building in the expertise of the electronics engineer and using the computer as an analogue device.

b. Computer simulation of a plasma

A plasma fusion experiment (hopefully the basis of a future generation of nuclear power stations) might contain 10^{20} particles (ions, electrons) at a temperature of 10^6 ^oC for time scales as short as 10^{-9} secs or as long as several seconds.

It is obviously extremely difficult to measure such high temperatures or such short times; in fact even setting up the experiment is usually an uncertain procedure - the plasma may touch a wall, vapourising it and hence producing contamination.

Since the basic laws governing the motion of the individual particles are known, it is possible to simulate a real plasma on the computer. A computer 'plasma' contains only say 10^4 particles, due to speed and storage restrictions, and since each computer particle is therefore representing around 10^{16} real ones, the force laws have to be modified to give realistic collisional effects. Further, ion-electron mass ratios have to be considerably reduced from the real values. Apart from these approximations, the computer plasma has great advantages to the physicist: the initial distributions, timescales, and energy growth rates, temperatures (measured by the degree of random movement) etc, are under perfect control.

We have found interactive computing to be of great assistance in the plasma simulation, both during the development stages of the simulation code, and also for exploratory investigations.

Firstly, when constructing the code, very many test runs are needed to check basic properties, for example that particles do not get lost; that momentum and energy are correctly conserved; and that timestep schemes are stable. The next stage is even more important, the physicist needs to investigate under what conditions, if any, the computer 'plasma' behaves like a real plasma. This involves a detailed study of ion and electron

oscillations; energy distributions; Fourier spectra; instability growth rates; initialisation methods and so on.

Graphical output is often required; simple plots may be output on the teletype; use of the display is of course preferable. Fig. 4 is a 'phase space' plot, of VX (velocity in X direction) against X. Fig. 5 is a velocity space plot, VY:VX. Many other diagnostics may be called upon.

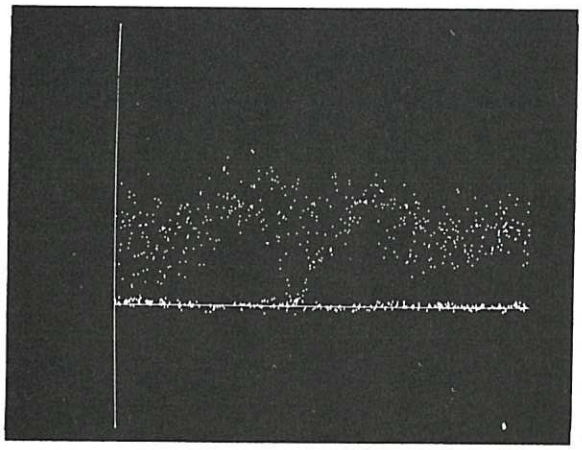

Fig. 4

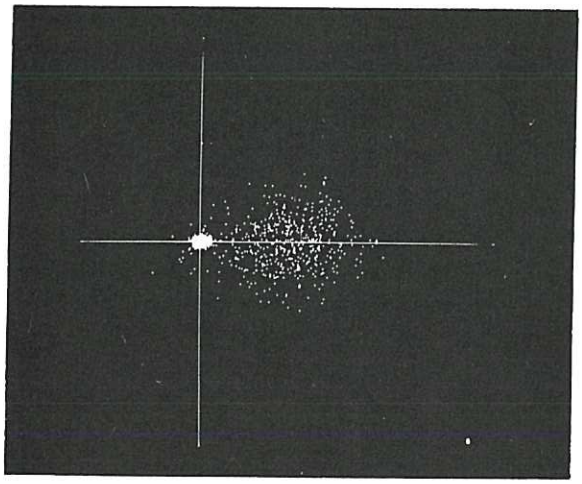

Fig. 5

It is fairly obvious that the use of interactive computing can give two advantages: many test runs can be completed in a very short time, giving rapid progress; and

the physicist can obtain a better 'feel' for the plasma behaviour than he could obtain with batch computing, as he can fully control the run, calling on relevant diagnostics as he sees fit. However, we would claim a further advantage, that of efficient computer utilisation. For when performing a series of exploratory tests, standard batch runs usually (in our experience) have a high failure rate. We estimate that as much as 75% of computer time may be wasted; besides the obvious total failures, many of the successful runs will have performed far more timesteps or iterations than was necessary. In contrast, interactive runs avoid complete failures, and curtail the run when the desired result is obtained.

c. Visualisation

The intense temperature of a plasma would instantly vapourise any material substance, so a plasma must be held well away from the walls of its container. This can be done by electromagnetic forces, usually applied by coil windings around the container.

Since plasma containment devices are often of toroidal shape, and the windings spiral around them, the configuration is very difficult for the draughtsman to draw, and for the designer to know the answers to such questions as: will the start and end of a winding match up correctly? will there be space between the windings for a probe in a specified position?

Mathematically however, the windings are usually very simple, typically being defined by just three or four parameters. So interactive graphics facilities can be used to assist, as follows. The windings are shown on the display for the set of parameters specified by the designer, who can view the winding from any position, and adjust the parameters at will. Unlike architectural applications, where hidden line removal for arbitrary (as opposed to rectangular) shapes is a difficult problem, the filamentary windings are best visualised by using intensity variations - that is, brightening near lines, and dimming distant lines. This is a very simple and effective process.

Fig. 6 shows a plan view of an unsymmetric toroidal winding. Fig. 7 shows a perspective view of a system representing cylindrical coils on a rectangular base. The 'menu' list provides both for the construction of standard shapes, and the possible scalings, rotations and perspectives. (These 'intensity variation' figures have lost

some of the variation effect in reproduction; they are reproduced from polaroid photographs which were taken by a hand camera from the 338 display. The DEC 338 allows 7 levels of intensity; all 7 were used on the originals).

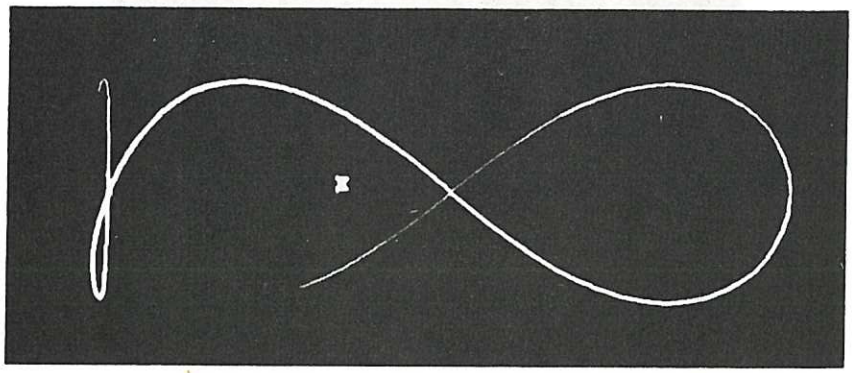

Fig. 6

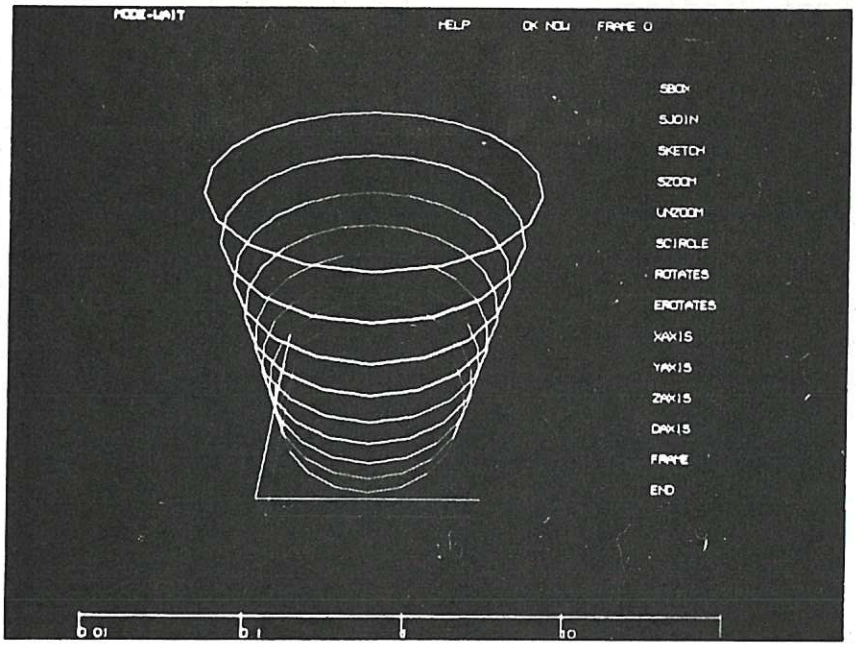

Fig. 7

Fig. 8 shows the contours of the field produced by a coil system; the positions of the coils are adjusted interactively so that the field acquires the desired properties - in this case, that plasma escaping from the centre region is led away into traps, so that it does not come into contact with the container walls.

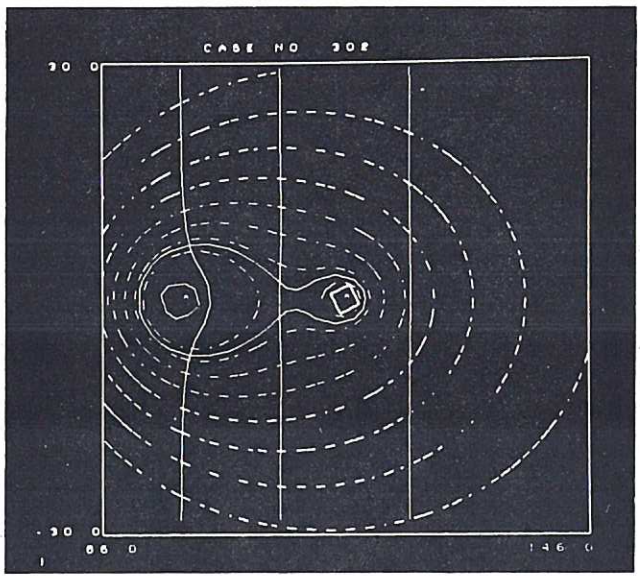

Fig. 8

d. Root Finding

Finding the roots of a function in the complex plane is a time-worn problem which frequently occurs in plasma physics - usually in investigations of the growth rates of plasma instabilities. Formerly, the engineer or physicist had to spend a considerable amount of time, performing tedious and error-prone algebra to locate the approximate position of the roots in which he was interested. Tentative application of a computer program would then yield "improved" estimates, perhaps only to find that a pathological region of the function gave rise to divergent behaviour in the numerical root-finding method. The turn round time for such a process would probably be of the order of several hours. Since present numerical techniques cannot fully automate the root-finding process, it is interesting to see what improvements the use of interactive computing can give.

There are essentially two different ways of approaching this problem, depending on hardware availability and suitability. Because a DEC 338 display was available on the KDF9, the first attempt was to use the powerful graphical output language (GHOST) to contour the function over specified regions of the complex plane. This not only gives useful topological information about the function but if the zero-height contours of the real and imaginary parts of the function are plotted on the same frame, the crossing points (if any) will give fairly reliable initial approximations to the required roots. (See Fig. 9).

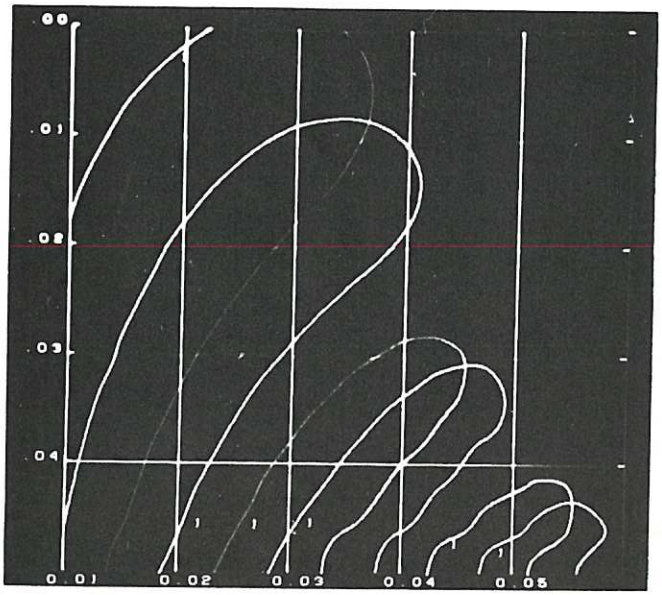

Fig. 9

These estimates can then be corrected by applying an iterative method - such as that put forward by Muller, which is simply based on a Taylor series expansion of the function about three neighbouring points. A useful offshoot from this approach is that the path of the iteration may be easily plotted, any divergence of the method being quickly detected.

The second version of the program, written for the ICL 4-70 machine, was considerably different since initially, fully interactive graphics was not available. The omission of the graphics lead to the use of much more sophisticated numerical methods to compensate for the lack of graphical information. Rather than try to apply a global numerical method, our experience suggested that a set of techniques should be available, the user being able to choose the most suitable for his particular function. In the present program, Muller's method is included as a general technique as it produces good results for reasonably behaved functions. It is supplemented by two other methods. One in which the function is locally approximated by a bilinear form thus overcoming the problem when a root is very near to a pole of the function. The second in which the Cauchy Residue Theorem is used to evaluate the number of roots in any given (circular) contour; the general form of the theorem then being used to explicitly evaluate the required roots.

Typically, the function not only depends on the complex variable z but also on a number of independent, real parameters resulting in the equation

$$f(z,\alpha_1,\alpha_2 \ldots \alpha_n) = 0$$

to be solved. Not only are the roots required for particular values of the parameters but also the position of the roots as a continuous function of any one of the parameters may be requested. This is achieved by predicting the new position of the root by quadratic extrapolation and using one of the above mentioned methods to refine the approximation. More sophisticated techniques have been developed for the Cauchy method in which step-halving and adjustments to the radius of the contour are automatically carried out to keep trackof the root; nevertheless, the most important ingredient is the experience of the program user.

e. Data Analysis

It is of great assistance to a research worker to have available all previously published information relevant to his work. Recently it was decided to produce a compilation of all the experimental data in the scientific literature on low-Z nuclear fusion cross sections. The data consisted of some 300 curves drawn from some 200 published papers, referring to some 50 different nuclear reactions. It was almost impossible to compare curves directly, as apart from simple scaling differences, curves could be drawn using linear-linear, log-linear, or log-log scales. The problem was to produce 50 figures, each including all the curves referring to a given reaction, drawn on a single standard scale; to renormalise or possibly reject obviously discrepant data, and finally to produce 50 publishable figures, with each curve and scale suitably labelled. The tediousness of such a task, if performed manually with the necessary precision, would have been prohibitive: in fact, the software was written in about eight man weeks, the curves were input during three days, and the figures for a preliminary publication were produced within a fortnight.

In brief, the typical operating procedure is as follows: a film strip projector or epidiascope is used to project an image of the material directly onto the phosphorescent screen of the DEC 338. (See Fig. 10). With the light-pen the user inputs to the display memory buffer two reference points defining the co-ordinate axes, followed by a sequence of points lying on each curve which he wishes to process. He then inputs to the PDP 8, by teletype, information about the scale (whether it is linear or logarithmic for example, and the co-ordinates of the two reference points).

Fig. 10

On the basis of this information, the KDF9 calculates the actual co-ordinates of each input point and computes a smooth curve passing through them, treating them as the knots for cubic spline interpolation. This curve is then displayed on the DEC 338, to permit a direct comparison with the original curve. The whole procedure takes perhaps one minute per curve (the actual computer time used being ~ 1 sec). If the user is dissatisfied with the agreement, he can then delete or add knots, again using the light pen. When he is satisfied with the fit, the co-ordinates of these knots, together with a name which he has assigned to the curve, are recorded in a previously specified file on the KDF9 disc. Almost any number of curves can be input in this manner, the only limit being the amount of disc space available. To recall this information, it is necessary only to specify the name of the curve and a suitable scale, which need not be identical with the original scale in any respect, for it to be displayed on the screen of the DEC 338. Any reasonable number of curves may be superimposed on the screen, the only limitation being the size of the DEC 338 memory buffer. Finally, if the user wishes a permanent record of a picture made up by superimposing curves in this manner, he can ask for the names of the curves and the scales to be recorded in an editorial file on the KDF9 disc, and the information in this file can be used to generate instructions on the magnetic tape belonging to the microfilm plotter, which will then provide a photographic record. The quality of this final output is designed to be such that it can be used directly for reproduction in the scientific literature, and it includes all necessary scale markings, captions and graticules.

f. Finite element mesh design

The method of finite elements has recently become popular for the solution of harmonic or biharmonic equations in field or stress analysis. The commonest element used is an equilateral triangle, and the first step in the process is the subdivision of the region of interest into a mesh of these basic triangular elements. See, e.g. J.K. Reid (5).

Near the edges of the domains, the triangles must be distorted in order to represent the boundary; but numerical analysis theory dictates that they must be as nearly equilateral as possible, as the error of the eventual solution depends inversely on the sine of the smallest angle. Another consideration in the triangularisation process is that the engineer or physicist will require more detailed information in certain areas, and hence will require a finer mesh there. Also, regions of maximum stress will require a finer mesh than unstressed regions to obtain acceptable overall accuracy.

We can see 3 solutions to this triangularisation problem.

a) entirely by hand. This is very tedious for large and complicated regions; it involves drawing all the triangles, adjusting to boundaries, providing finer meshes where indicated, and avoiding small angle triangles. In addition, the co-ordinates of the mesh points must then be read off and fed into the computer.

b) entirely by computer. The process can be automated; starting with a uniform triangular mesh covering the whole region, exterior triangles are omitted; corners of boundary triangles are moved onto the boundary, and some maximisation process juggles the triangles to avoid small angles. There are two difficulties with this process; first, although a fairly simple computer program can correctly handle say 99% of triangles, it is extremely difficult to guarantee 100% success for every shape of region; and attempting to cope with every possibility always produces a complicated and lengthy code. Secondly, the engineer or physicist must specify the finer mesh regions by some unnatural process, such as supplying functions which are negative inside the desired area, zero on the boundary and positive outside.

c) using interactive graphics. This can combine the best of both above methods. A simple code performs the initial triangularisation, getting say 99% of it correct, and projects its attempt on the display. Fig. 11 shows a simple

hole-in-plate example, which has been well triangulated by the simple program. The engineer/physicist can now 'touch up' any defects by moving corners with the light pen, using the 'menu' keywords ZOOM (zooms in on a specified part of the picture); MOVE (moves an indicated node to an indicated position), and UNZOOM (returns to normal view). To refine the mesh is equally simple; the keyword REFINE is given, and a triangle interior indicated. This triangle is then subdivided by the program into 3 triangles.

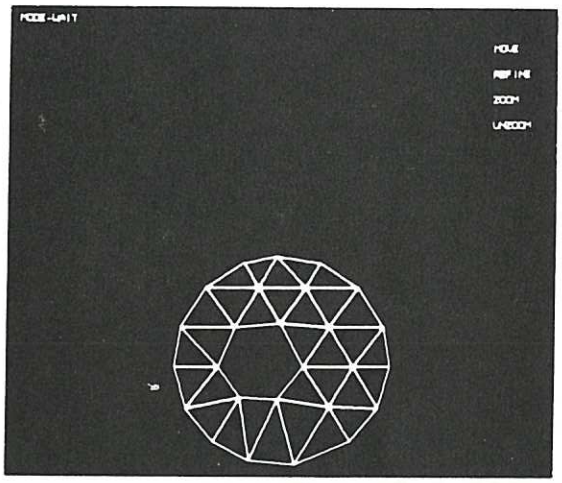

Fig. 11

The final result of a single node adjustment, plus some additional refinements, is shown in Fig. 12. The mesh is now ready for presentation to the numerical part of the program to obtain the desired field or stress solution.

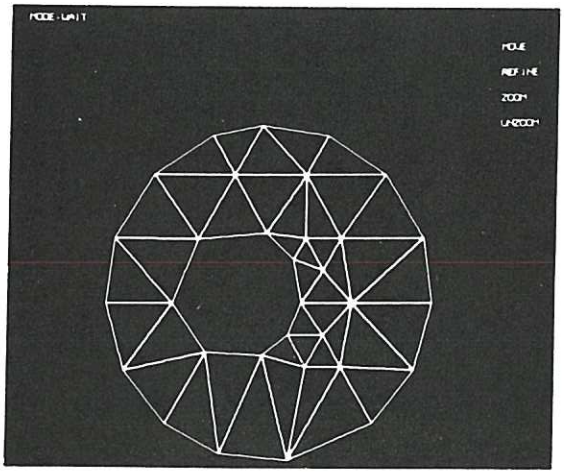

Fig. 12

4. Interactive programming techniques, and graphics requirements

To tackle interactively the problems described in this paper, we adopt the following procedure (we almost always use FORTRAN):

1. All parameters of the problem are initialised to their most common values.

2. Then, and after each stage, the program returns with the prompt KEYWORD?

3. Associated with each of the parameters is a 'keyword' which enables the variable to be identified. For example, the timestep in an integration, DT, having a default value of 0.1 is changed to 0.05 by replying DT 0.05 to the KEYWORD? prompt. The program recognises the command by reading DT into WORD, and 0.05 into VALUE, and then testing the keyword (DT) against the complete list of keywords in a series of 'IF' tests, until it reaches IF(WORD.EQ.'DT') DT=VALUE.

4. Keywords are also associated with program actions. For example, 'START' might perform a series of timesteps or iterations using previously defined data. The relevant IF statement here may be

```
IF(WORD.EQ.'START') GO TO 7
```

This type of program control is very easy to write in FORTRAN and to some extent is self-documenting if sensible names for the keywords are chosen. By using this method, most programs can easily be converted to interactive operation and can then, by sensible use of prompts, be run by non-expert computer users. Also, when the user wishes to run such an interactive program as a standard off-line job, the data file for the run will consist of the same data as would be used for interactive control and will be self-documenting because of the 'keyword' system.

Interactive programs are usually more dependent on graphics than the average as information must be conveyed quickly and concisely to the user. We see three special requirements for interactive graphics:

i) since roll-in, roll-out streams are of necessity small, the graphics must be as efficiently coded as possible to allow maximum space for the users program.

ii) also, since the computing time slice is usually very short, the graphics must be as fast as possible.

iii) if a refresh display is used, the size of the refresh buffer is often a severe limitation; graphical information must be packed into it as efficiently as possible. (Storage tubes of course do not have this problem).

These requirements favour the use of a low-level graphics package, built round the specification of the relevant device. Unfortunately, this usually means the introduction of a new low-level language; however it is possible by careful design to combine a standard user-image with efficient device utilisation; in the GHOST system this is done by the provision of special processors.

5. Conclusion

We consider that the problems met in a scientific environment can be classified into three groups:

i) Problems - usually of a routine nature - that are best done by conventional batch computing;

ii) Difficult or exploratory problems that cannot be formulated for purely automatic computing; they require constant interaction from the problem originator, and hence the interactive approach is almost essential;

iii) Problems that can benefit to varying extents from interactive computing, possibly just in the program development stage, possibly throughout.

We consider that the greatest benefit obtainable from interactive (as opposed to batch) computing is not the possible time savings - useful though these may be - but the improved understanding that a problem originator can obtain by interactive study of problems of the second type; a study that requires him to be an expert in his problem field, not a computer scientist.

6. References

(1) P.C. Poole, 'Some aspects of the Egdon 3 operating system for the KDF9',IFIP Congress, Edinburgh, 1968.

(2) 'A Users Guide to Cotan', UKAEA Paper, Culham Laboratory Report CLM-R75.

(3) Larkin, F. M., 'A Graphical Output Language and its Implementation' UKAEA Paper, Culham Laboratory Preprint CLM-P139.

(4) Prior, W.A.J. 'The GHOST Graphical Output System User Manual', UKAEA Paper, Culham Laboratory Report CLM-PDN 8/71.

(5) Reid, J.K. 'On the construction and convergence of a finite element solution of Laplace's equation' J. Inst. Math. Applics. 9, 1-13, 1972.