NUMERICAL SIMULATION OF

NON-NEWTONIAN FLUID FLOW

IN

MIXING GEOMETRIES

by

STEPHEN PAUL HAVARD B.Sc.

Thesis submitted to the C.N.A.A. in partial

fulfilment of the requirements for the

degree of Doctor of Philosophy

The collaborating establishment was

Baird and Tatlock Ltd.

Department of Mathematics and Computer Science

The Polytechnic of Wales

Mid. Glamorgan CF37 1DL

September 1989

NUMERICAL SIMULATION OF

NON-NEWTONIAN FLUID FLOW

IN

MIXING GEOMETRIES

by

STEPHEN PAUL HAVARD B.Sc.

Thesis submitted to the C.N.A.A. in partial

fulfilment of the requirements for the

degree of Doctor of Philosophy

The collaborating establishment was

Baird and Tatlock Ltd.

Department of Mathematics and Computer Science

The Polytechnic of Wales

Mid. Glamorgan CF37 1DL

September 1989

CERTIFICATE OF RESEARCH

This is to certify that, except where specific reference is made, the work presented in this thesis is the result of the investigation undertaken by the candidate.

Candidate .....

Director of Studies .....

DECLARATION

This is to certify that neither this thesis or any part of it has been presented or is being currently submitted in candidature for any other degree other than the degree of Doctor of Philosophy of the C.N.A.A.

Candidate S.P. Howard

Acknoweldgements

Firstly I wish to acknowledge my sincere gratitude to my director of studies, Dr R.W. Williams and to my supervisors Dr R.G. Miles and Dr D.G. Knight for their encouragement, guidance and constructive criticism during this research project.

In addition to this, a special thanks is particularly extended to Dr R.G. Miles for his partnership in writing various papers.

Secondly I am grateful to S.E.R.C. for their financial support during the last three years.

And finally I would like to express my appreciation to my parents for their support and especially to my wife, Mari for her understanding, patience and constant encouragement during the period of accomplishing this thesis.

Numerical Simulation of Non-Newtonian Fluid Flow in Mixing Geometries

by

Mr. S.P. Havard B.Sc.

SUMMARY

In this thesis, a theoretical investigation is undertaken into fluid and mixing flows generated by various geometries for Newtonian and non-Newtonian fluids, on both sequential and parallel computer systems.

The thesis begins by giving the necessary background to the mixing process and a summary of the fundamental characteristics of parallel architecture machines. This is followed by a literature review which covers accomplished work in mixing flows, numerical methods employed to simulate fluid mechanics problems and also a review of relevant parallel algorithms.

Next, an overview is given of the numerical methods that have been reviewed, discussing the advantages and disadvantages of the different methods.

In the first section of the work the implementation of the primitive variable finite element method to solve a simple two dimensional fluid flow problem is studied. For the same geometry colour band mixing is also investigated.

Further investigational work is undertaken into the flows generated by various rotors for both Newtonian and non-Newtonian fluids. An extended version of the primitive variable formulation is employed, colour band mixing is also carried out on two of these geometries.

The latter work is carried out on a parallel architecture machine. The design specifications of a parallel algorithm for a MIMD system are discussed, with particular emphasis placed on frontal and multifrontal methods. This is followed by an explanation of the implementation of the proposed parallel algorithm, applied to the same fluid flow problems as considered earlier and a discussion of the efficiency of the system is given.

Finally, a discussion of the conclusions of the entire accomplished work is presented. A number of suggestions for future work are also given. Three published papers relating to the work carried out on the transputer networks are included in the appendices.

CONTENTS

CHAPTER 5 MATHEMATICAL SIMULATION OF AXISYMMETRIC MIXING PROBLEMS

CHAPTER 1

INTRODUCTION

This thesis presents research work which was undertaken into the numerical simulation of flows generated by the mixing of fluids.

1.1 Background

Mixing is a widespread and very important process and as yet only a very limited amount of theoretical study has been given to it, possibly because the nature of the mixing problems encountered in industry are rheologically complex. Analysis of the phenomenon of mixing has been carried out by considering each case on its own merits, and so far generalisation has been difficult. Stirring in vessels is an essential part of many industries such as the polymer processing, food processing, water, petroleum, chemical and biochemical industries. Examples from within these industries include the mixing of resins, adhesives, paints, pharmaceuticals, fertilizers and the fermentation of broths. Much use is made of the mixing process in the home, for example the manual stirring of sauces, also food mixers, blenders and processors are used to aid in the making of sponges and souffles, and the liquidisation of soups.

Mixing is usually achieved by rotating an impeller in a fluid until it reaches a required level of homogeneity. A necessary requirement of the system is for the power consumption to be minimised without degrading its performance; this can be achieved by carefully choosing the factors affecting blending, such as the best impeller geometry for the fluid properties, the shape of the containing vessel and the impeller speed. The calculation of the

power consumption for various mixer geometries can be used to compare the efficiency of different systems.

Consider a simple mixing problem where a fluid is enclosed in a container and a propeller is attached to a vertical shaft which is used as the agitator, and is rotated inside the fluid (see fig. 1.1).

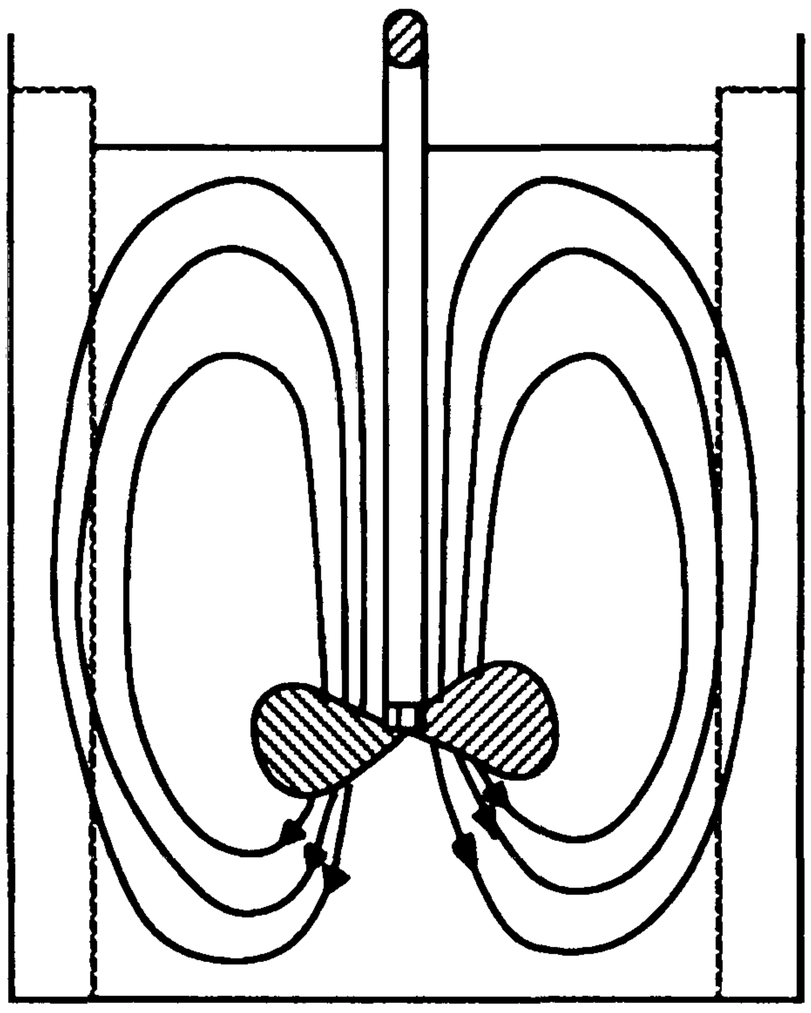


Figure 1.1 Axial flow generated by a propeller.

The flow pattern generated inside the stirred vessel is the only factor which affects the quality of mixing. What is seen to happen is that liquid streams move at high speeds near the impeller, which then circulate outwards towards the fixed outer walls of the vessel where the movement of the fluid is either quiescent or slower, and so the slower moving fluid replaces the space produced by the fast moving fluid. Thus the momentum generated by the axial circulatory motion enables the flow to reach the very extremities of the vessel and continues until the fluid is mixed to the required consistency.

For this type of geometry and for a particular class of fluid (highly shear-thinning, as described later) with the impeller operating at high speed it is possible for a cavern of turbulent flow to occur near the impeller while the remainder of the flow regions remain stagnant, producing a poor overall mixing. A similar agitator is a disc turbine where the flow pattern produced by it is radial (fig. 1.2).

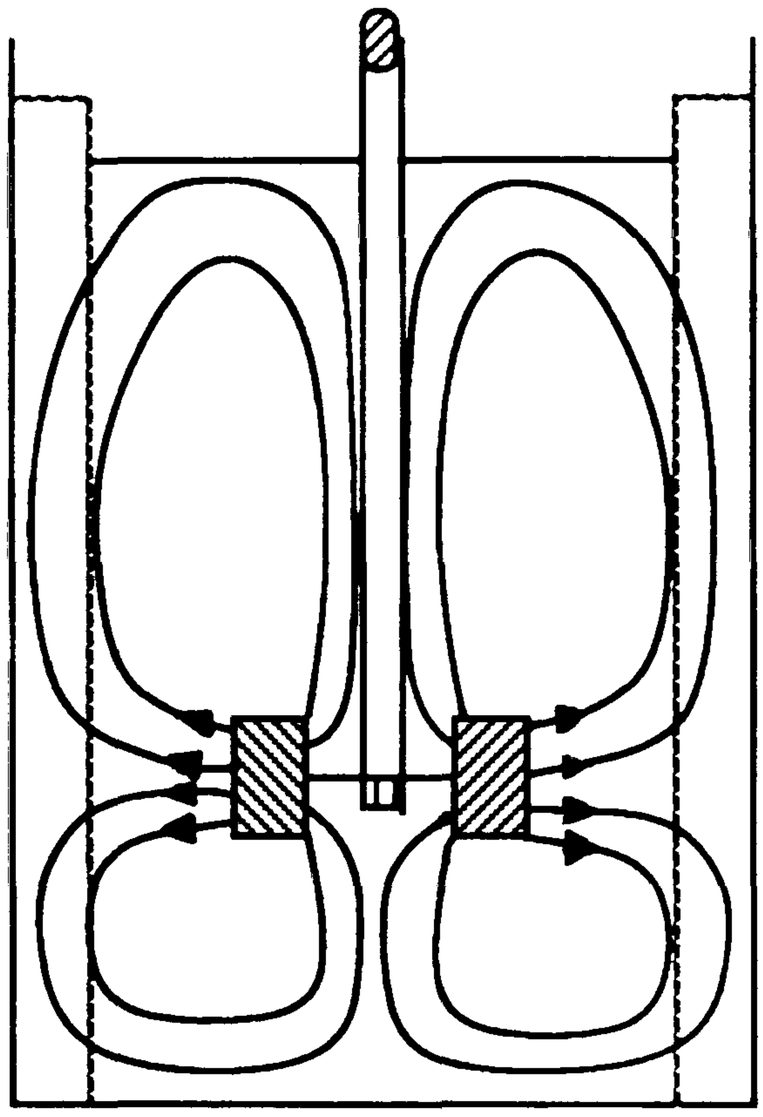


Figure 1.2 Radial flow generated by a rotating disc turbine.

Baffles, usually attached to the walls of the vessel, are employed to prevent the phenomenon of gross vortexing occuring when low viscosity liquids are stirred in a vertical container with a centrally mounted stirrer. Normally baffles are not required with high viscosity liquids since gross vortexing is not a problem.

There are three categories of flow in mixing :

(i) Re < 10; low values of Reynolds number. (Here Re is the abbreviation for Reynolds number, which represents the speed of the flow.) This is known as the laminar region, where the mixing is initiated by the velocity distribution in the vessel and by molecular diffusion. This very slow mixing is suitable for processing viscous liquids.

(ii) 10 < Re < 1000; medium to high speed. This is known as the transition or streamline region which occurs at higher values of Reynolds number. In this region the factors affecting the flow ,i.e. velocity distribution in the vessel and molecular diffusion are considerably more significant, causing the flow produced to show a greater degree of instablity.

(iii) Re > 1000; very high Reynolds numbers. In this region, known as the turbulent region, the flow breaks down and becomes chaotic. Mixing occurs quickly and efficiently because of the movement of the turbulent vortices.

For the studies carried out in this thesis the types of flow considered were restricted to orderly non-turbulent flows.

Of the physical properties of the fluid which affect the flow, viscosity is normally the most important. This is defined as the property of a fluid to resist the flow through internal forces and molecular attraction. It is also worth noting that the amount of energy used to perform the mixing process increases with the viscosity of the fluid.

1.1.1 Types Of Fluid

The classification of liquids falls into two main categories :

1.1.1.1 Newtonian Fluids

The viscosity of these fluids are independent of time and are unaffected by any applied forces. Examples of such fluids are syrup, water and glycerol.

1.1.1.2 Non-Newtonian Fluids

This is the most common type of real fluid for which the viscosity is affected by either time or applied forces. These fluids are categorised below according to their attributes:

(i) Shear-thinning fluids, also known as pseudoplastic fluids, where the fluid viscosity decreases as rate of shearing is increased. For this reason impellers which operate at high speed will have a low viscosity in the vicinity of the impeller and high viscosity outside this region. Examples of these fluids are liquid food products, pharmaceuticals, wall paper paste, soaps and detergents.

(ii) Shear-thickening fluids, also known as dilatant fluids, where the fluid viscosity increases as shear rate is increased. For this type of fluid the apparent viscosity is higher near the impeller than it is elsewhere in the container. Examples of these fluids are china clay slurries, starch suspensions, cornflour and custard.

(iii) Plastic fluids. This type of fluid has the property known as yield stress. These materials act as solids under small stresses and as fluids after a critical value of the stress has been exceeded; for example liquid abrasive cleaners such as Jif, and toothpaste.

(iv) Thixotropic fluids, whose apparent viscosity decreases with respect to time whilst the shear rate is kept constant. The stirring of these fluids at a fixed rate can result in a significant lowering of the viscosity with respect to time. Eventually the fluid will return to its initial viscosity when the stirring has terminated. Examples of these are tomato ketchup, salad cream and non-drip paints.

(v) Anti-thixotropic fluids, also known as rheopectic fluids, where the apparent viscosity is slowly increased with respect to time for a constant shear rate. This very uncommon phenomenon is known as rheopexy, examples of such behaviour are an alkaline perbunan latex solution and some vinyl plasticides.

(vi) Viscoelastic fluids. Such fluids are capable of exhibiting both viscous properties and elastic properties depending on the conditions under which the fluid is subject. They usually resemble a gel-like substance which when subjected to certain conditions will flow. Examples of these include egg white, shower gels, blu-tack, silly putty and "slime".

In this thesis only Newtonian and shear-thinning liquids will be considered.

1.1.2 Types Of Stirrer

To ensure that the mixing process is performed efficiently for the many fluid types, of such differing characteristics, a number of different mixers have been empirically designed. The range over which an impeller is suitable depends largely on the viscosity of the fluid. The majority of impellers are centrally mounted in a vertical vessel. There are three main categories of impellers used in industry:

(i) Impellers with a small surface area blade which operate at high speeds and depend on good momentum transport for mixing. These types of impellers operate best under turbulent flow conditions for low and medium viscosity liquids, where the flow patterns produced are mainly axial or radial. Examples of this type of small impeller are the six bladed Rushton turbine and marine type propellors (fig 1.3) and are used in conjunction with the following types of liquids; oils, resins and glues.

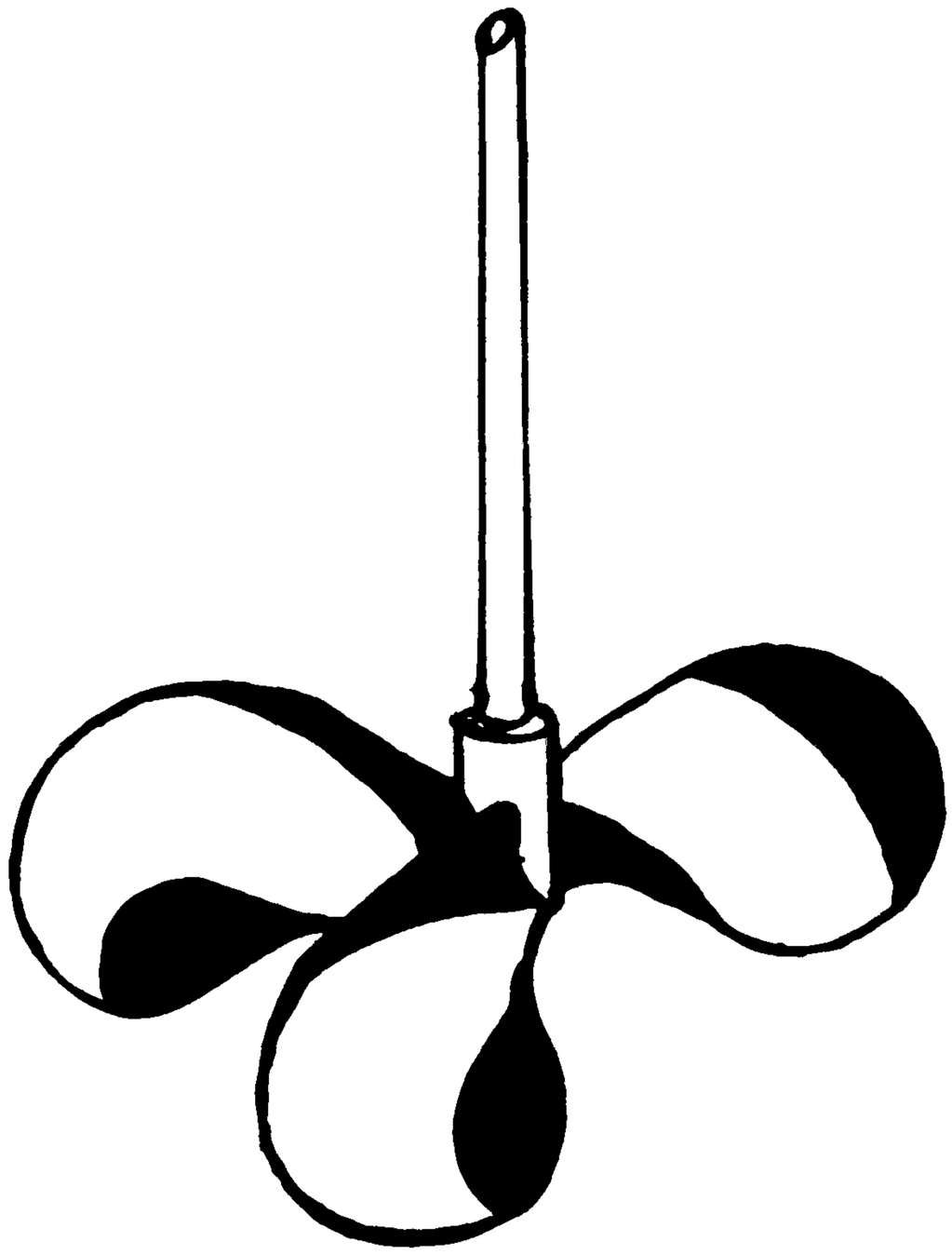


Figure 1.3 a Three-bladed propellor

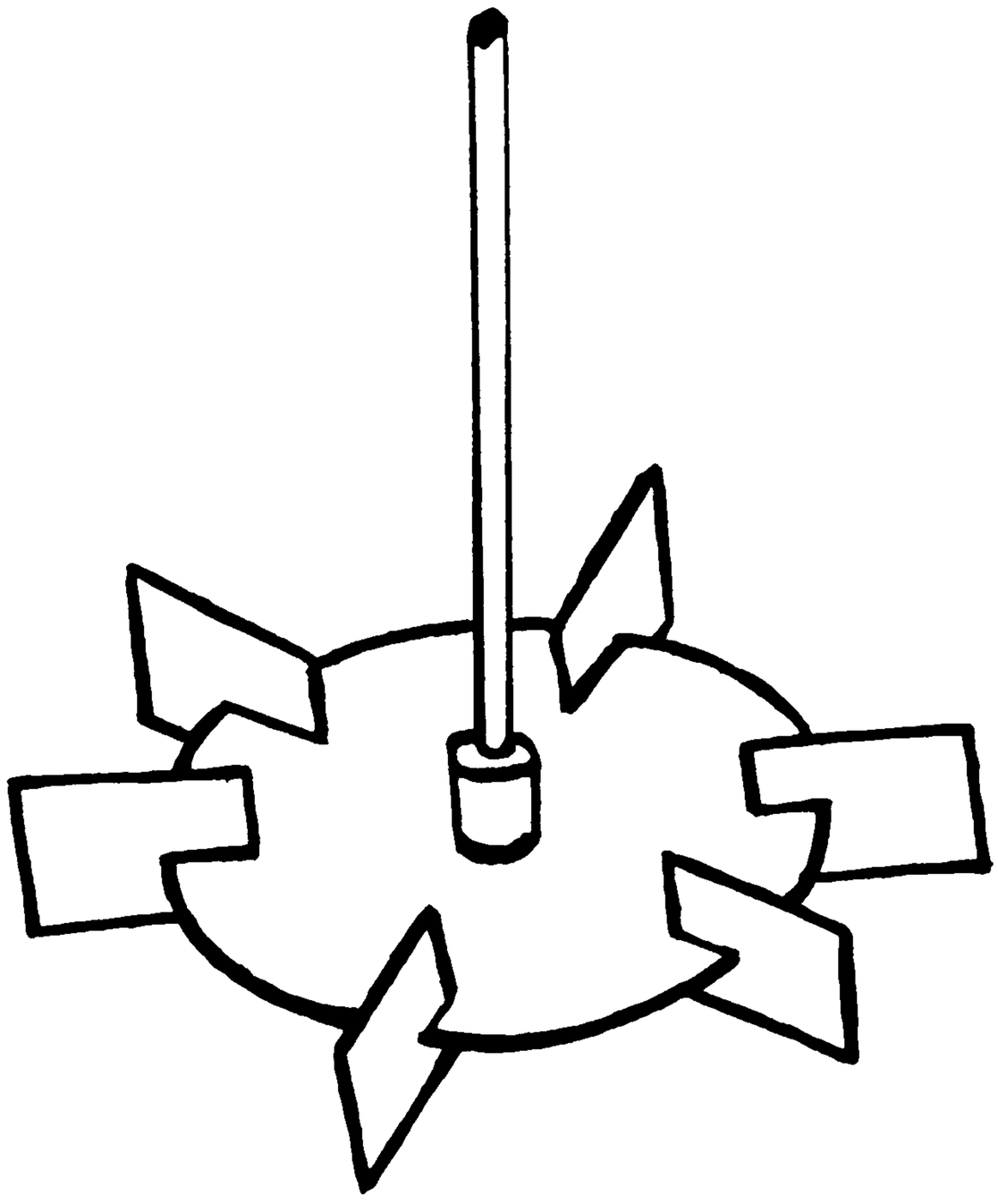


Figure 1.3b Six-bladed disc turbine.

(ii) Impellers with a large surface area blade which operate at slow speeds. These impellers produce an extensive flow and minimise the stagnant regions because of the shape and size of the impeller which is such that it passes through the very extremities of the vessel. The flow is mainly rotational and lacks any flow in the axial direction. Examples of this type of impeller are the anchor and gate impellers (see fig. 1.4 of a anchor impeller) and are applicable for high viscosity fluids e.g. syrups.

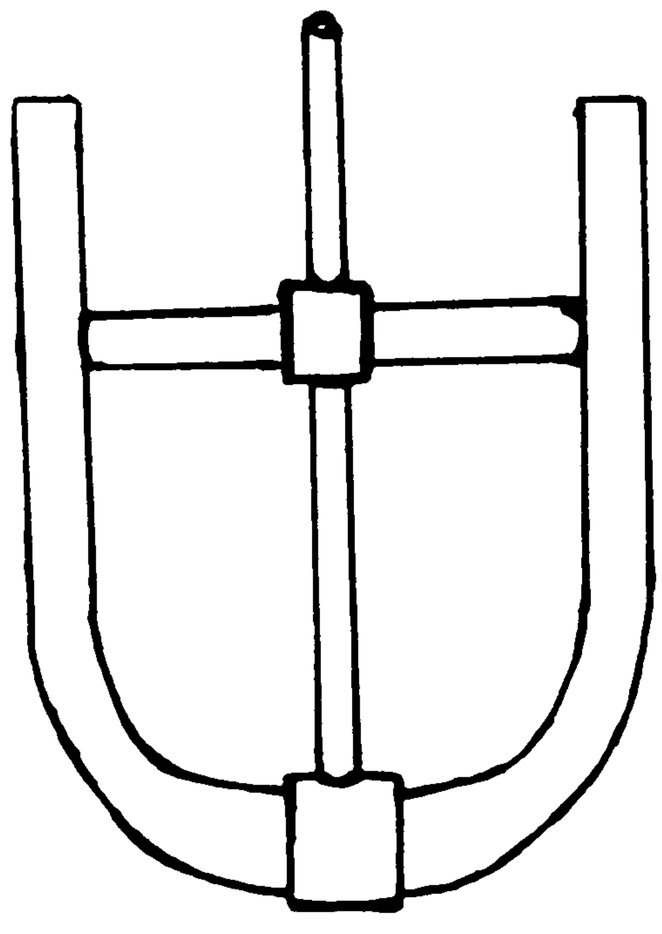
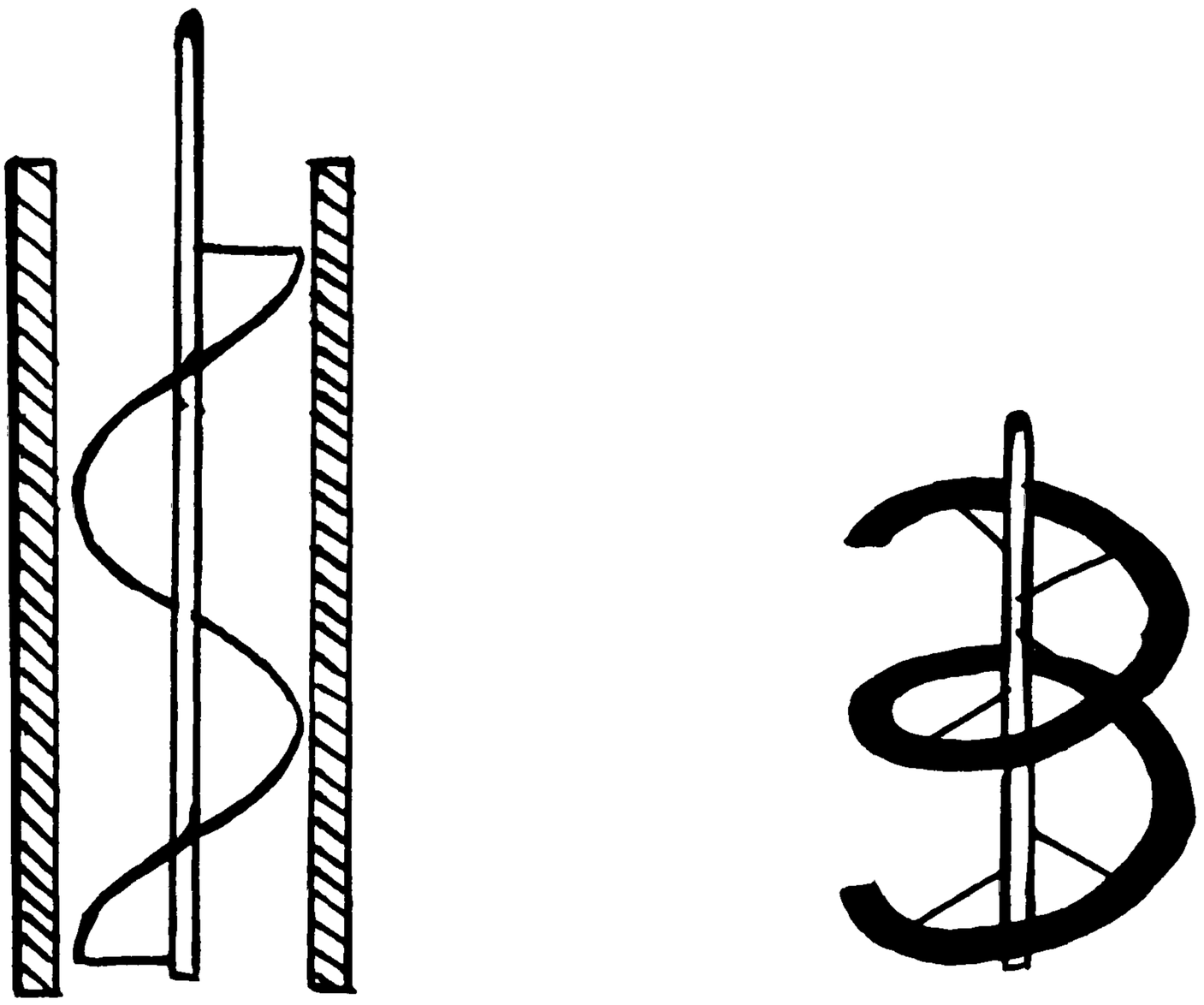


Figure 1.4 Anchor impeller

(iii) More complex impellers which again have a large surface area and operate at low speeds are the helical screw which has a large clearance between it and the vessel wall and the helical ribbon with only a small clearance at the wall of the vessel (see fig. 1.5). These types of flow are composed of both rotational and axial contributions which is similar to a pumping action and so enables every part the vessel to be reached. These rotors are also suitable for mixing fairly high viscosity liquids such as stiff pastes and putties.



(a) (b)

Figure 1.5 (a) Helical screw with draft tube

(b) Helical ribbon

For further information regarding impellers and their applications reference the following classic texts [40,78,85,108].

1.2 Aims Of This Research Work

In accomplishing this research work we hope to gain a better understanding of the flows and mixing in stirred vessels. We have only considered the mixing phenomenon from a theoretical standpoint, as much experimental work on flow fields and mixing has already been carried out in this area by Bodalia [6] and others [103,78,40,85]. The experimental and theoretical results obtained by Bodalia are used as a comparison with the theoretical results obtained later in this report.

The theoretical analysis of mixing flows is very complicated, as the geometries for the problem can be very complex, (for example, gate type anchor agitator). When considering problems involving non-Newtonian fluids, rather than Newtonian fluids, there is an added degree of difficulty due to the non-linear structure of the fluid. However, since there remains much speculation regarding the mathematical modelling of such fluid flows, in complex geometries, it was decided in this project to concentrate on simpler geometries and fluids, in the hope that if satisfactory results were obtained for these simple mixing sytems then these could be used in the future to develop more complex geometries and fluids.

The purpose of mathematical models is to predict experimental behaviour and to simulate experimental results. When applied to flow problems, these models can be used to predict the kinds of flow and various paramters about the flow under consideration, for example the degree of mixing and power consumption. These models also allow for scale up or scale down, which is useful for comparison with other systems; scale up being used mostly to predict flows in industrial mixing from benchtop arrangements i.e. they can be used to model 'real life' industrial applications. This can result in a considerable reduction of time and also lead to large financial savings as it avoids the necessity of carrying out much experimental work.

Initially a two dimensional flow problem was considered for development work of numerical procedures. The geometry was that of a plate moving over an infinitely long cavity, with the fluid being contained inside the cavity. This was chosen as it is one of the standard test geometries for which much research work has already been carried out. It is also a far simpler problem than the full three dimensional mixing problem. The numerical technique chosen to solve the flow problems presented in this report is very complicated to implement and so by studying a simpler problem initially, overcoming any difficulties that might arise with the method, before proceeding to tackle the more realistic three dimensional problems. The numerical method adopted is the primitive variable finite element approach.

Investigational work was also carried out for colour band mixing for the same cavity geometry, showing how a band of colour disperses through the fluid with time until a required consistency is reached.

We then progressed to three dimensional flow problems. The first of these was that of the flow obtained when a finite cylinder is rotated in a fluid contained in an outer stationary cylinder. After obtaining a numerical solution for this problem we considered two more complicated geometries; in the first instance a disc rotor replaces the cylinder rotor and secondly the rotor is replaced by a simple version of the Rushton turbine.

Again dispersive mixing of a band of coloured tracer is investigated for the cylinder and disc rotor geometries.

The results obtained for both sets of problems were compared as far as was possible with experimental and theoretical values obtained by other researchers in the field.

It was our intention on reaching this stage in the project to consider flows produced by viscoelastic liquids. Before investigating such a complicated fluid system it became apparent that a faster computer was necessary than the one available at the Polytechnic of

Wales. This led us to explore the possibilities of speed-up provided by parallel architecture machines, and the remainder of the research time was spent in designing and implementing an efficient parallel algorithm to solve the problems mentioned in the earlier parts of this section with a view to eventually modifying this algorithm to incorporate viscoelastic fluids (excluding colour transport).

Until recently the design of the computer has been based on the Von Neumann architecture, these are known as sequential systems where an ordered set of instructions are processed sequentially on a single data stream. Outstanding progress has been achieved in improving the performance and miniaturisation of the components used in these systems. The rate of advancement in performance levels is now much slower due to the nature of the difficulties faced by computer designers becoming more and more complex.

For this reason computer designers have turned their attention to parallelism as a means of speeding up the computer. Initially they remained with single processor machines, introducing a degree of parallelism by adopting a number of effective methods, for example interleaved memory, pipelining, vector processing and "very long instruction word" (VLIW) to achieve a significant improvement in performance and efficiency.

This naturally led to the consideration of parallel architecture machines. Multi-processor systems have a variable number of processors linked together in a network, and are classified into two main categories:

(i) Single instruction stream/ multiple data stream (SIMD), in this type of system the same single stream of instructions is broadcast to every processor in the network, each processor accessing its own personalised set of data. At any point during execution of a program on such a system each processor either performs exactly the same instruction as the rest or does nothing at all. Examples of such systems are the ICL DAP with up to 1024 processors and the Connection Machine with 65,536 processors.

(ii) Multiple instruction stream/ multiple data stream (MIMD), in such a system seperate instruction processes and associated individual data sets are shared among the processors, with each processor working independently, performing its own set of tasks on its own data set. Communication between processors is possible via message passing. Examples of such systems are Intel iPSC, Transputer networks and the Ncube.

An important aspect of parallel systems is the way in which the processors are linked together as this dictates how the instruction/data reaches the relevant processor. In some cases the number of processors a message has to pass through to go from a particular processor to another can be very long. For this reason consideration has been given to discover the best way in which the processors in the network can be linked together. Examples of such constructions are the ring, 2-D array and binary hypercube. This aspect becomes most significant when a large amount of communications takes place between distant processors.

In the case of parallel architecture computers two type of memory systems have been employed, shared memory and local memory systems:

(i) In shared memory systems all the processors in the network have access to a common memory system. This gives every processor the ability to access any part of memory, as a large number of processors may wish to access the same data locations simultaneously precautions are taken to guarantee that no processor is allowed to read from the memory location until an appropriate write to the location by another processor has completed. There are three main categories of memory scheme which have been used:

(a) The memory system is composed of a number of memory units and each processor in the network is linked directly to each memory unit.

(b) A common bus or channel is set up between the processors in the network and the memory units, such a bus will work very slowly if a large number of messages are passed to and from memory, as only a limited number of processors can use the channel at the same time.

(c) The processors are linked to the memory units via a number of switching boxes, each box has bi-input and bi-output links. This in effect enables each processor to have a direct link to each memory unit, with the added advantage of having a significantly reduced number of communication lines compared with that of the approach mentioned in (a).

(ii) Local memory systems also known as distributed memory architectures have a memory unit associated or attached to each processor, which is only accessible by that processor. Information can only be passed around the network by message passing between processors, no other processor can directly access the memory which is associated with another processor.

The writing of software and the designing of algorithms that will best exploit the parallel architecture of the machine is posing a new and exciting challenge to programmers. Algorithms which were efficient on sequential machines may prove to be disappointing when implemented on a parallel machine, and algorithms which were previously inefficient on sequential machines may perform well on parallel machines due to the inherent parallelism in the algorithm.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

For this thesis a survey of the current literature available in the field of mixing and numerical methods was undertaken. It was seen as important to obtain an overview of mixing, to determine what work had been undertaken and to find the best numerical techniques to use in our research work. Much time was spent in choosing the numerical method and in finding literature which best explained it in the context of fluid flow problems and also in justifying its preference over rival methods. It was also important to find experimental and theoretical results by other researchers which could be used to compare with our results so as to justify the numerical method used and evaluate the results obtained.

Decisions had to be made concerning what type of computer system to use for the latter work and as one of the systems considered was a parallel architecture machine, based on transputers, a review of parallel algorithms relating to the finite element method was undertaken. Included also in our survey is a list of relevant introductory texts to parallel programming, and in particular to Occam and transputers.

In this chapter we present a summary of the literature that has been reviewed. The work covered in this review falls into the following categories :

i) Non-Newtonian liquids and theoretical/experimental work on mixing

ii) Appropriate numerical techniques

iii) Two-dimensional Cavity driven flow problem

iv) Three-dimensional Mixing flow problems

v) Parallel computing and algorithms

2.2 Non-Newtonian Liquids and Theoretical/Experimental Work on Mixing

To gain familiarisation with real life non-Newtonian liquids and their relevance to industry we found a good introductory text by Barnes [2]. For mathematical representations of non-Newtonian liquids and their applications Walters [114], Bird et al [5], and Van Wazer [109] were most useful. These books give the grounding which is necessary for the understanding of the inherent non-linear ramifications of non-Newtonian mixing problems.

Many books have been written on the subject of mixing. Two recent books give a very full up-to-date review of the whole topic. The first of these is a book edited by Harnby, Edwards and Nienow [37], in which there is an interesting chapter on mechanically-agitated vessels, presenting the different types of agitators and their uses. Edwards presents a chapter on the mixing of low viscosity liquids in stirred vessels, covering items such as power input, flow patterns, turbulence measurements and mixing time. There is also a chapter on mixing of high viscosity fluids by J.C. Godfrey who covers points such as mixer application, mixing rate and power consumption.

The second book, which is edited by Ulbrecht and Patterson [108], covers the mixing of viscous non-Newtonian liquids, showing the flow patterns for such liquids. Also discussed is power consumption and mixing efficiency.

Other older classic books which deal with mixing are Holland[40], Nagata [78], Oldshue [85] and others [18,48], which mention the many types of impellers, and also the categorising of the different types of fluids by their properties.

There are also a number of papers available on the subject of mixing. Early reviews of mixing were carried out by Quillen [95], who presented the major kinds of mixers available in the context of their application. Rushton [100] in his paper gave a review of the field of mixing , considering topics such as the mixing of liquids, pastes and solids. The mixing of elastic liquids are described by Ulbrecht [106] and more recently by Collias and Prud'homme [17].

An analysis of mixing phenomena has been carried out by Spragg et al [103]. In their paper they investigated theoretically steady laminar flow and mixing for a centrally mounted disc rotor in a cylindrical vessel containing fluid. Results were given for Newtonian and non-Newtonian fluids. Dispersive mixing of bands of marked fluid were also performed showing a sequence of concentration fields with respect to time.

Recently much experimental and theoretical work has been carried out by Bodalia [6] on the flow fields due to simple stirrers such as cylindrical and disc rotors rotating in cylindrical vessels containing Newtonian and non-Newtonian fluids. Also discussed in this thesis are power consumption and discharge efficiency.

With the advent of high speed compact computing systems, the numerical prediction of fluid flows has become easier. This has reduced the reliance on the previous practice of basing results on experimentation alone. Cross et al [20] in their paper which discusses future trends for software engineering in computational fluid dynamics (CFD), review a number of CFD based software engineering packages of varying levels of sophistication. They are based either on the finite difference method or the finite element method but are not immediately applicable to us.

Lohner [66] in his paper reviews the way forward for finite elements in CFD. Firstly he points out a number of areas in which finite element methods are presently inadequate and then makes suggestions as to how these can be overcome, for example unstructured multigrid methods and adaptive refinement schemes.

Another very recently available package Polyflow [93] which is designed to solve most types of fluid flow problems and is based on the finite element method. It has a user friendly preprocessing routine which allows the user to interact with the program and also to visually see the mesh being generated. It also has a postprocessing routine to produce graphical output of the results that are required. The package can deal with most regular geometries either 2-D or 3-D and for Newtonian and most types of non-Newtonian fluids.

This package is very general in its application, however it has only just become available. In the meantime we had developed our own software which performs the calculations required and also has sufficient flexibility.

Recently turbulent flow studies through the mechanism of chaos have been undertaken. One prominent book is Ottino [87] who summarises recent work beautifully in his book.

Theoretical partial differential equations have been developed to model the flow of many fluids for various mixing geometries. Analytical methods are generally restricted to straightforward geometries containing Newtonian or simple non-Newtonian fluids under slow flow conditions. Otherwise, the mathematical equations describing the flow are very complex and numerical methods are required. In what follows references for analytic solutions are presented, the references being ordered according to the speed of the flow (i.e. by the Reynolds number).

2.2.1 Fluids flowing with Re -> 0.

Flows in this region are categorised as primary flow. In text [114] analytical methods are applied to solve a number of Newtonian fluid flow problems where the results are presented in terms of fluid velocities and couple. Jeffrey [55] employes analytical techniques to model the flow produced by ellipsoidal shaped agitators rotating in Newtonian fluids.

For the case of non-Newtonian fluids analytical approaches are adopted by Van Wazer [109] to solve for flow between infinite length concentric cylinders and by Borne [7] for flow between concentric spheres. They also calculate values for the couple involved.

2.2.2 Fluids flowing with Re < 10.

Much analytical work has been carried out in the streamline range. Waters and King [120] and Waters and Gooden [119] encountered many difficulties in solving the flow of elastic fluids around ellipsoidal shaped stirrers. Fosdick and Kao [32], Thomas and Walters [105], Walters and Waters [116], Walters [115] and Wein [121] have solved the the flow of an elastic liquid between two concentric spheres and they also present theoretically calculated couple values. Flows due to a cone-and-plate geometry were solved by Giesekus [33] and Walters and Waters [117]. The complicated analytical formulae resulting from these analyses require computer evaluation in most cases.

2.2.3 Fluids flowing with 10< Re <1000.

No analytical work was found within this range.

2.2.4 Fluids flowing with Re > 1000.

A number of non-Newtonian fluids have been modelled analytically in this range. Mitchka and Ulbrecht [74] considered the flow of power law fluids and inelastic fluids about rotating cones and spheres whilst Kale et al [60] and Williams [123] adopted a 'Von Karman' flow approach between parallel plates[112]. Chaotic turbulent flow may nowadays be simulated by simple mathematical methods e.g. Ottino [87].

2.3 Appropriate numerical techniques

To simulate flows for higher reynolds numbers and for complicated geometries and fluids numerical methods and computers are required. There are two main numerical methods used to solve fluid flow porblems, which are :

i) The Finite Difference Method (FDM)
ii) The Finite Element Method (FEM)

2.3.1 The Finite Difference Method

There is much literature available on the finite difference technique. Smith [102] provides a good introductory text to the technique as applied to all types of problems, including certain simple fluids problems. A book outlining a large number of techniques employed for the solution of Newtonian fluids is Roache [99]. Crochet, Davies and Walters [19] is a comprehensive text which is essentially aimed at incompressible fluid flow problems, for various types of non-Newtonian fluid.

2.3.2 The Finite Element Method

The literature available for this approach is also extensive. An introductory text to the method is given by Davies [22] which considers a varied range of general problems with many examples, however incompressible fluid flow problems are not covered by this book. Reddy [97] explains in simple terms many of the difficult concepts behind the finite element method. He also covers useful topics such as mesh generation and mesh refinement, as well as two dimensional incompressible fluid flow and time dependent problems. An useful text which covers all aspects of the finite element method is Zienkiewicz [129].

There are a large number of numerical solvers which are currently available for the solution of systems of equations. In the case of small linear problems simple direct solvers are used, for example Choleski decomposition for symmetric matrices. When dealing with large linear sparse symmetric matrices, it is a customary practice to employ a bandwith reduction scheme first, for example the Cuthill-Mckee algorithm [21], before using a special band matrix solver.

For very large scale problems involving many thousands of variables computer storage becomes a problem and therefore either the frontal method is applied [42] or special sparse/band matrix solvers are used, for example iterative schemes such as successive over relaxation (SOR) or preconditioned conjugate gradient method.

Our interest lies mainly in the frontal method which is a standard tool in finite element analysis and was originally devised by Irons [53] for the solution of symmetric sparse matrices. The frontal scheme has undergone a number of modifications, for example the scheme was modified by Hood [41] to treat unsymmetric matrices with symmetric sparsity pattern and by Duff [25,26] who treated the unsymmetric patterns.

Reid [98] suggested a multifrontal scheme involving substructuring the finite element problem and initialising a number of frontal matrices where the assembly process is controlled by a tree data structure. Duff and Reid [27] applied the multifrontal scheme to solve unsymmetric sets of linear equations. The crux of the scheme is to build up an elimination tree to control the procedure. Duff [28] suggested that the elimination tree could serve as a schedule for parallel computation and in [29] implemented a scheme on a shared memory multiprocessor.

When considering the finite element method applied to incompressible fluid flow problems there are number of standard texts. Firstly there is the book by Crochet, Davies and Walters [19], already mentioned, which considers the finite element method applied to fluid flow problems for Newtonian fluid, then generalised Newtonian fluids and finally

visco-elastic fluids. Secondly there is the book by Hughes and Taylor [42] which describes in detail a solution procedure for the Newtonian Navier-Stokes equations.

In a recent paper Marchal and Crochet [68] describe a new mixed finite element approach for calculating viscoelastic flows at very high values of Deborah numbers for both Maxwell-B and Oldroyd-B fluids.

For advection dominated flows for high enough Reynolds number the solutions obtained are often corrupted by spurious oscillations occuring because of a lack of diffusion present in the numerical method. These oscillations can be overcome by employing one of the many upwind schemes available. The basic upwind method is described in Wait and Mitchell [113]. Brooks and Hughes [11] have carried out much work on upwind schemes and in their paper they present a new upwind method which overcomes the difficulties which are associated with other classical upwind schemes. As a continuation of this work Hughes et al [43,44,45,46] have recently published a series of papers in which consideration is given to the problem of sharp internal/boundary layers. Much mathematical analysis has been carried out on this upwind scheme in the following papers [56,57,58,59,81].

We have chosen to adopt the finite element method as it is a far more versatile method than its rival finite difference method.

2.4 Two-dimensional cavity driven flow problem

2.4.1 Experimental Studies

Chien et al [14] have recently presented some experimental results for laminar mixing in several kinds of 2-D cavity flows, with the Reynolds number ranging from 0.1 to 100. For studies of streamline visualisation at high Reynolds numbers there are papers such as these by Winters and Cliffe [128] and Peyret and Taylor [92]. Ottino[87] has carried out several experiments on slot mixing.

2.4.2 Theoretical Flow Studies

Cliffe et al [15] discuss flow modelling and use of the finite element method to solve for the primitive variables. Their paper considers four problems, one of which is driven flow in a square cavity. Results are presented for Re=1 and 400. Another paper, by Peeters [91] also solves the same problem using a finite element approach and presents a streamline plot at Re=400. For higher Reynolds numbers there are papers by Dhatt [23] for Re=1000, Mizukami [75] for Re=1000 and 10000, and Nasser et al [80] also at Re=1000.

Bozeman and Dalton [9] use various implicit finite difference approximations of the Navier-Stokes equations to solve the steady two-dimensional flow in rectangular cavities, presenting streamline plots at Re=100 and 1000. For cavity flow Nallasamy and Krishna Prasad [79] employ finite difference techniques to obtain solutions over a wide range of Reynolds numbers from 0 to 50000. Ottino [87] has also presented mathematically generated chaotic/turbulent data on slot for comparison with experimental results.

2.5 Three-dimensional Flows In Mixing Problems

2.5.1 Fluids flowing with Re -> 0

The results of the numerical solution for the primary flow of a Newtonian fluid for a number of rotating stirrers are presented in terms of various parameters such as fluid velocities, shear rates and couple [6,35,36,64,65].

Bodalia [6] considered flow produced by rotating cylinder and disc impellers for both Newtonian and non-Newtonian fluids. For the case of Non-Newtonian fluids Wein [121,122] and Williams [126] found solutions for flows produced by rotating ellipsoidal and disc impellers. Numerical methods were used by Griffiths and Walters [36] for flows induced by rotating cones, Keentok et al [61] for flows produced by vanes and Paddon and Walters [89] who modelled the flow produced by rotating cylinders.

2.5.2 Fluids flowing with Re < 10

Bodalia [6] considered the flow induced by rotating cylindrical and disc stirrers for both Newtonian and variable viscosity fluids. The numerical modelling of elastic fluids around rotating discs was carried out by Griffiths et al [35]. The flow of elastic fluids in the cone-and-plate geometry was solved numerically by Paddon [88,89]. Work was undertaken by Williams [125] on flows around finite rotating discs, cylinders, spheres and also thin discs [127].

2.5.3 Fluids flowing with 10< Re< 1000

A number of researchers have considered flow problems for both Newtonian and Non-Newtonian fluids in this range. Numerical techniques have been employed to obtain solutions for flows generated by rotating disc stirrers for both types of fluid (Bodalia [6], Dijkstra and Van Heijst [24], Spragg et al [103] and Wang and Yang [118]). The numerical solution of the flow of fluids in a container with a rotating cover or disc was undertaken by Pao [90], Bartela and Gori [4], Nirschl and Stewart [83] and Kramer and Johnson [63]. The problem of the flow in a container with a rotating lid was also considered by Lugt and Abboud [67] showing the effect of axisymmetric vortex breakdown with and without temperature effects at Reynolds numbers greater than 1000.

If the height of the concentric cylinders is large with respect to the gap between the cylinder and outer wall, the flow is termed Couette flow. For Reynolds numbers greater than some critical value a phenomenon known as Taylor vortices occurs. In a paper by Stuart [104] he presents a thorough investigation of the experimental and theoretical evidence concerning the Taylor vortex phenomenon. Other papers discussing this phenomenon include Neitzel [82], Mullin and Lorenzen [77] and Cliffe and Mullin [16].

2.5.4 Fluids flowing with Re > 1000

For flow in this range only a small amount of work has been undertaken. Pao [90] used numerical methods to solve for flows in the disc geometry whilst Hyun [49] solves for flows in the cylinder geometry. Issa and Gosman [54] used the finite difference method to compute three dimensional two-phase flows in mixer vessels. Hutton [47] looked at current progress in the simulation of turbulent incompressible flow by the finite element method. Ottino [87] has also considered mathematical chaotic/turbulent mixing applications suitable for real life mixing situations at high Reynolds number range. Flows in this range are of no interest to us in this thesis.

2.6 Parallel Computing and Algorithms

Designing efficient parallel algorithms is an active area of much current research. Ortega and Voigt [86] review the current state of numerical methods for the solution of partial differential equations on vector and parallel systems. They mention the attributes of these computers and explain how to choose algorithms which take advantage of them. The aim of the book is to present suitable algorithms for parallel and vector computation and also to show deficiencies which occur either due to the hardware or the algorithm employed. Discussion on the concepts relating to the design of parallel algorithms can be found in both Quinn [96] and Modi [76]. With regard to the finite element method a number of different approaches have recently been suggested for the design of parallel algorithms. These are all on-going research programs.

Amestoy and Tilch [1] employ the finite element method to solve the compressible Navier-Stokes equations, where the algebraic system produced by the finite element formulation is solved using the multifontal technique which is implemented on a shared memory multiprocessor system.

Williams and Glowinski [124] manipulate irregular meshes on a distributed memory multiprocessor machine. The algorithm incorporates an incompressible Navier-Stokes solver with a three stage implicit time step. A preconditioned conjugate gradient approach is employed for a linear system whilst Newton's method is adopted if the problem is non-linear.

Most other work in the design of parallel algorithms for the finite element method has been applied to the analysis of structures. The nature of the equations are quite different but an understanding of how efficient parallel algorithms have been implemented will help to give a more complete comprehension.

An element-by-element solution scheme has been developed by Barragy and Carey [3] on a shared memory system. Farhat and Wilson [31] suggest substructuring the domain and present two parallel solution schemes, direct and iterative, to solve the algebraic system produced. This is implemented on a local memory system. Another local memory solution scheme is suggested by Nour-Omid et al [84] who employ a substructuring of the mesh technique and the solution process is a hybrid scheme based on direct triangulation and preconditioned conjugate gradient procedures. In a purely theoretical paper Melhelm [70] suggests a solution technique which is based on ideas taken from both frontal methods and band matrix solvers.

In our work we have made use of a local memory MIMD system, based on a network of transputers. The most efficient programming language for the transputer is Occam and is the language we have employed in our work. This is based on computing sequential processes which is described in a book by Hoare [39]. An excellent text on how to program in Occam has been written by Kerridge [62]. Other useful texts on this subject are [8,10,12,13,94]. A series of books produced by INMOS relating to all aspects of transputers and Occam are also available [38,50,51,52].

CHAPTER 3

NUMERICAL SIMULATION IN FLUID DYNAMICS

3.1 Introduction

Numerical methods are employed to closely approximate the solution of a problem. By applying suitable analysis numerical methods will give solutions that are as accurate as the data permits.

There are two main numerical methods used in fluid flow problems, these are:

i) the finite element method,

ii) the finite difference method.

The numerical scheme which has been used to solve the problems described in this report is the finite element method. There are a number of disadvantages in using the finite difference method as opposed to the finite element method :

(a) it is awkward to specify boundary conditions along curved boundaries.

(b) it is awkward to deal with complex irregular geometries because of the limited amount of flexibility possible using structured grids.

(c) it is difficult to accommodate local mesh refinement.

(d) it is difficult to deal with non-rectangular meshes.

(e) it is straightforward to write the code for the finite difference formulation for a particular geometry but not so easy to alter it for a different geometry.

The finite element method overcomes the above difficulties even though it is far more costly in machine time than the finite difference method, but this may be acceptable when traded off against the disadvantages outlined above. In what follows the finite difference method is mentioned and then a fuller explanation of the finite element method is given.

3.2 The Finite Difference Method

The finite difference method is a well known, much tried and tested, numerical technique which has been used to solve partial differential equations with applied initial/boundary conditions for certain geometries. The finite difference formulae are derived from Taylor's series, where derivatives defined at specific points are approximated by difference quotients over a small interval. A fuller description of the method applied to fluid dynamics problems can be found in classic texts by Roache [99] and Crochet, Davies and Walters [19].

3.3 The Finite Element Method

In this section we introduce and explain the basic concepts of the finite element method. A fuller treatment can be found in the books of Zienkiewicz [129] and Crochet, Davies and Walters [19]. The finite element method is based on a piecewise application of a variational technique.

The variational method involves replacing a differential equation by its equivalent integral form. In applying the variational or weak formulation the order of the derivatives of the dependent variable in the integral form of the equation can be lowered by transferring differentiation from the dependent variable to the test function and thereby relaxing the continuity requirements on the solution. In variational formulations boundary conditions fall naturally into two categories, natural and essential conditions.

3.3.1 The Rayleigh-Ritz Method

This method provides an algorithm for minimising a given functional and requires that a suitable complete set of linearly independent basis functions ϕ_j with constants C_j be chosen. The exact solution is approximated by a linear combination of the trial functions where constants C_j are chosen to minimise the functional I(U) which can sometimes be interpreted in terms of the total energy of the system under consideration. The problem reduces to a system of linear algebraic equations, which then need to be solved [22].

3.3.2 Method Of Weighted Residuals

For the Rayleigh-Ritz method to be applied it is required that a functional be found which satisfies a minimum principle, but unfortunately for the majority of physical problems such a principle does not exist. Where such a functional cannot be found other techniques of numerical solution have to be employed. These are known as weighted residual methods.

3.3.2.1 The Galerkin Method

In this method the inner product of the residual weighted with the chosen basis functions, is set to zero. It is possible with an even ordered differential operator to obtain the Ritz method from the Galerkin method. If a weak formulation is allowed then the continuity condition on the approximating functions can be lessened as differentiation is transferred from the solution to the weighting functions.

It is for nonlinear problems, as in our case, in which the differentiability on the dependent variable can be weakened, that the Galerkin method is most suitable [19,22,97]. Therefore, it is the Galerkin approach which is used to solve the fluid problems in this report. The Rayleigh-Ritz method is used to obtain the stream function from primary variables.

The variational method suffers from a number of drawbacks, for example it would be impossible to obtain an approximating function that would satisfy complicated essential boundary conditions or cover a complex geometry. There is also no general systematic method of generating these functions for a given differential equation.

3.3.3 Formulation Of The Finite Element Method

The finite element method is a powerful general and systematic numerical method which overcomes the difficulties associated with the variational approach. The stages to be performed in applying the finite element method can be summarised as follows, and the advantages of the finite element method over the variational approach are clearly apparent:

a) the domain of the problem, however complex it may be, can be subdivided into a finite set of non-overlapping simple shaped elements connected together at a finite set of points which are called nodes. In regions of great change the mesh may be refined whilst allowing for large elements in quiescent regions.

b) the governing equations are formulated by a Galerkin (or Rayleigh-Ritz) formulation for the discretized form of the problem.

c) the approximating function is systematically constructed for each element incorporating the essential boundary conditions and satisfying the continuity criterion imposed by the variational formulation.

d) the application of the method produces a set of algebraic equations in terms of the dependent variables defined at the nodes. Each node is only affected by the nodes directly associated with it, having no links with distant nodes. This produces a large sparse/banded matrix.

e) these equations are then solved to give values for the dependent variables.

3.3.3.1 Types Of Elements

Our aim in this section is to consider the different approximating functions which can be defined in terms of piecewise polynomials for the elements. The number of terms in the polynomial must be equal to the total number of degrees of freedom of a particular dependent variable associated with the element, otherwise the polynomial will not be unique.

The standard types of elements used in fluid mechanics are triangles and quadrilaterals. Finite element meshes can be composed of either triangles or quadrilaterals and it is also a long-standing practice to construct meshes which are a mix of both element types. For example it is better to use triangular elements near a curved boundary, as it is possible to closely fit any curve by triangles, and the interior of the domain can be covered by quadrilaterals.

There also exist more elaborate elements which so far have not been applied to fluid dynamics problems, and therefore will not be considered here.

A first-order bi-linear polynomial for an element with 3 nodes and 1 degree of freedom at each node is sufficient to define an approximation over the element. This element would obviously be a triangular element, with the nodes being defined at the vertices, (fig 3.1) and the approximation is:

$$U(x,y) = C_a + C_b\, x + C_c\, y \qquad (3.1)$$

Another polynomial which is also bi-linear but has four linearly independent coefficients is (3.2). This type of polynomial would cover a four noded element, such as a quadrilateral, where the nodes are specified at the vertices (fig. 3.2), and now:

$$U(x,y) = C_a + C_b x + C_c y + C_d xy \qquad (3.2)$$

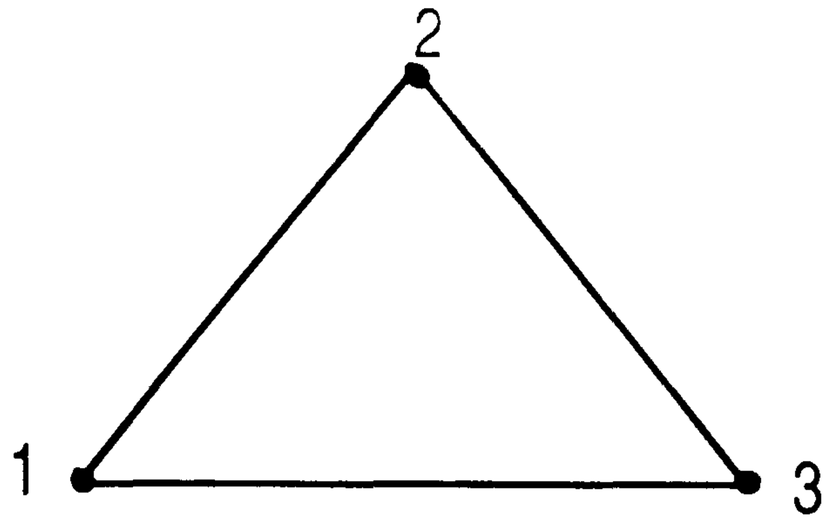


Figure 3.1 Three-noded triangular element

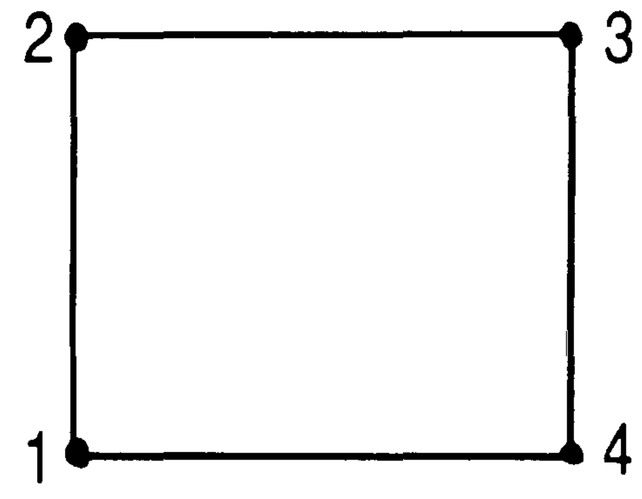


Figure 3.2 Four-noded quadrilateral element

When considering second order polynomials a very large choice of elements are available, however only two will be considered. The bi-quadratic polynomial:

$$U(x,y) = C_a + C_b\, x + C_c\, y + C_d\, xy + C_e\, x^2 + C_f\, y^2 \qquad (3.3)$$

which is identified by means of six coefficients covers a triangle with nodes defined at the vertices and the mid-sides of the element, fig. 3.3. The 2nd order polynomial defined by 8 linearly independent coefficients:

$$U(x,y) = C_a + C_b x + C_c y + C_d xy + C_e x^2 + C_f y^2 + C_g x^2 y + C_h x y^2 \qquad (3.4)$$

satisfies the requirement for the 8 noded quadrilateral element, known as the serendipity element, where the nodes are specified at the vertices and mid-side points of the element (fig. 3.4).

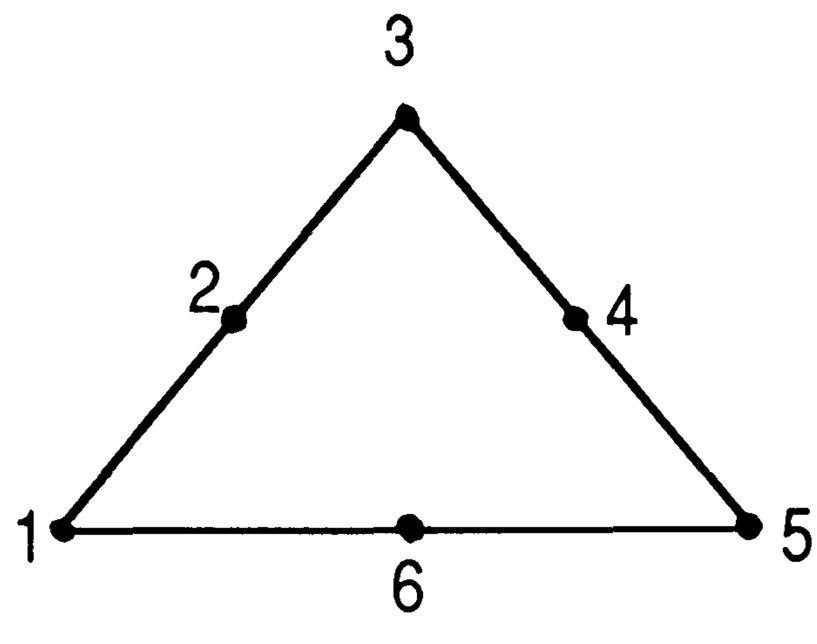


Figure 3.3 six-noded triangular element

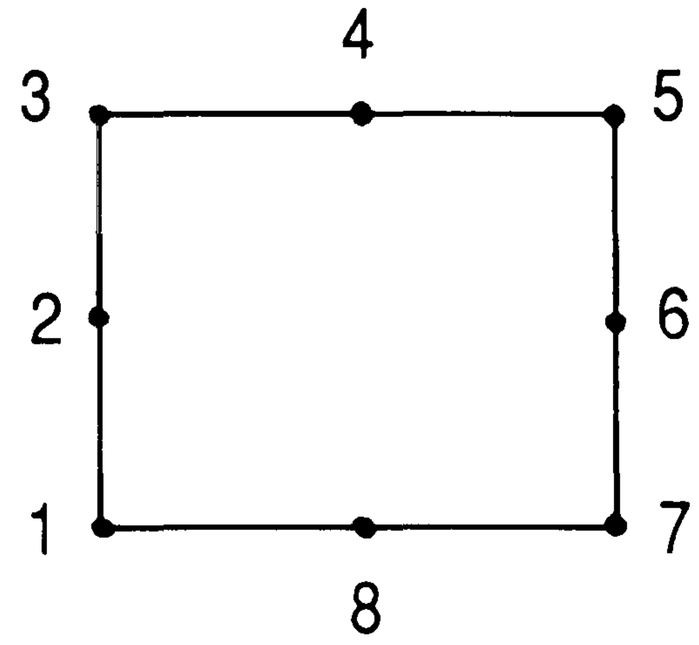


Figure 3.4 eight-noded quadrilateral element

3.3.3.2 Isoparametric Transformation

Isoparametric elements are very important in finite elements as they enable curvilinear boundaries to be accurately represented. However, constructing shape functions for this type of element result in very complicated algebraic expressions, which make integration extremely difficult. It is preferable to integrate over some standard region.

The mapping onto a standard region is achieved by performing an isoparametric transformation from this curvilinear element to a simple element of standard form, see fig 3.5. This is accomplished by a coordinate transformation. Mapping from the global x-y plane onto a local ξ–η plane.

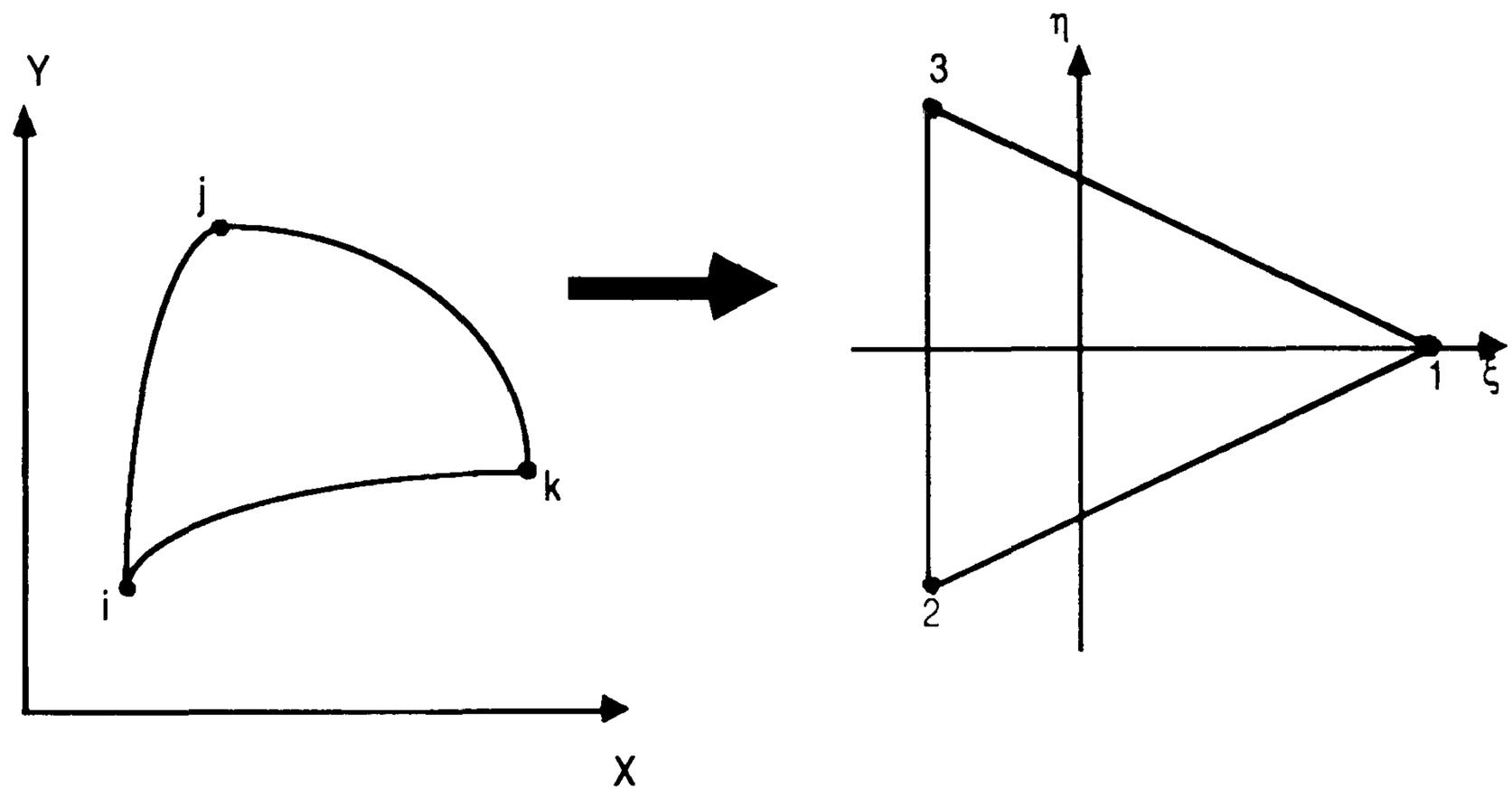


Figure 3.5

This mapping is performed on all elements in the mesh as it is far easier to work with elements in their local form and then transform them back at a later point. For reasons associated with the subsequent numerical integration it proves convenient to normalise the coordinates. The shape functions for the three noded general triangular element take the simple form:

$$N_1(\xi,\eta) = 1/3\ (\ 1 + 2\xi)$$

$$N_2(\xi,\eta) = 1/3\ (\ 1 - \xi - 3\eta) \qquad (3.5)$$

$$N_3(\xi,\eta) = 1/3\ (\ 1 - \xi + 3\eta)$$

where $-1/2 \le \xi \le 1$ and $-\sqrt{3}/2 \le \eta \le \sqrt{3}/2$

The transformation between the global and the local coordinate systems is

$$x = \sum_{k=1}^{m} N_k(\xi,\eta)x_k \quad ; \quad y = \sum_{k=1}^{m} N_k(\xi,\eta)y_k \quad , \tag{3.6}$$

where x_k and y_k are the abscissae of the element nodes and $N_k(\xi,\eta)$ are the m shape functions in terms of the local coordinates.

Derivatives are easily converted from one coordinate system to another by

$$\begin{bmatrix} \frac{\partial}{\partial\xi} \\ \\ \frac{\partial}{\partial\eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial\xi} & \frac{\partial y}{\partial\xi} \\ \\ \frac{\partial x}{\partial\eta} & \frac{\partial y}{\partial\eta} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x} \\ \\ \frac{\partial}{\partial y} \end{bmatrix} = \underline{\mathbf{J}} \begin{bmatrix} \frac{\partial}{\partial x} \\ \\ \frac{\partial}{\partial y} \end{bmatrix} \tag{3.7}$$

where $\underline{\mathbf{J}}$ is the Jacobian matrix. The determinant of this matrix is used in the transformed integrals e.g. as follows:

$$\iint dxdy = \iint |\det \underline{\mathbf{J}}| \, d\xi d\eta \quad . \tag{3.8}$$

The integration has to be performed numerically using numerical quadrature. Hence the integral becomes

$$\iint f(x,y)dxdy = \sum_{j=1}^{n} W_j f(x_j,y_j) \quad , \tag{3.9}$$

where the (x_j,y_j) are the abscissae of the n sampling points and W_j the associated weights. Transforming into the local coordinate system, the resulting quadrature formula is

$$\iint f(x,y)dxdy = \sum_{j=1}^{n} Wj|J(\xi_j,\eta_j)|f(x_j,y_j) \tag{3.10}$$

where x_j and y_j are given by applying (3.6).

3.3.3.3 Imposing Essential Boundary Conditions

The Dirichlet boundary conditions can be imposed in one of two ways :

(i) The method of biasing entries in the system stiffness matrix corresponding to the boundary nodes. The leading diagonal of the entry is augmented by some large number, and the corresponding entry on the right hand side is given a similar scaling.

(ii) The method of altering the entries in the system stiffness matrix corresponding to the boundary nodes. The leading diagonal of the entry is replaced by one, and the remaining entries in the row set to zero. The boundary value is inserted into the respective position on the right hand side.

3.3.3.4 Numerical Solvers

For fairly small linear problems simple solvers can be used such as Choleski decomposition for symmetric matrices. When considering very large linear problems involving sparse symmetric matrices it is advisable to try and reduce the bandwidth of the matrix by employing a bandwidth reduction algorithm, before solving using a special band matrix solver. An example of such a band reduction technique is the Cuthill-Mckee algorithm [21].

For large nonlinear problems, either some form of linearisation is performed or Newton's method is applied and the solution proceeds iteratively using Gaussian elimination at each stage to solve the matrix equation. When dealing with a few thousand variables computer storage becomes a problem and therefore either special sparse/band matrix solvers are used or the frontal method is applied [42]. The frontal method has become a standard tool in finite element analysis as it achieves a significant saving in computer time and storage over the standard Gaussian elimination approach and is the method that we have adopted in our work.

3.4 Different Approaches of The Finite Element Method for Steady Flows

In this section a brief summary is given of the three main approaches of the finite element method applied to incompressible fluid flow for both Newtonian and Non-Newtonian (variable viscosity) fluids.

3.4.1 Stream Function-Vorticity Approach

The governing equations of the problem can be re-formulated from the natural U,V,P variables into vorticity-stream function formulation and this type of formulation is frequently adopted in finite difference methods. The equations are solved for these basic variables from which the velocities and pressure can subsequently be calculated.

It should be noted that in choosing this type of formulation it has the associated problem of defining vorticity boundary conditions. For this reason the method is seldom used with finite elements.

3.4.2 Velocity - Pressure Model (U,V,P Model)

In this approach a solution is sought in terms of the primitive variables-velocities (U,V) and pressure (P). Once a solution has been obtained for the primitive variables then the stream function, vorticity and tractive forces may be obtained.

The finite element model is non-positive definite as zeros appear on the leading diagonal of the overall stiffness matrix, which is due to the pressure being evaluated from a continuity equation. Therefore only equation solvers which use some sort of pivoting strategy can be used.

It will be seen from the equations of motion, that pressure is one order less in differention than the velocities and therefore the approximating functions used should be one degree

less for pressure relative to the velocities. This means that pressure is not defined at every node and this increases the complexity of implementing the model on the computer.

3.4.3 The Penalty Model

The penalty method is designed to overcome the problems involving pressure in the U,V,P model, by eliminating pressure from the equations. This formulation treats the continuity equation as a constraint among the velocity components, which causes the pressure term to drop out of the integral form of the governing equations. The method has been applied mainly to Newtonian fluid mechanics problems, with only a very small number of researchers who recommend its use in numerical non-Newtonian fluids mechanics [19].

3.5 The Application of the Finite Element Method to Time Dependent Problems.

For time dependent problems the standard approach taken is to adopt a two step formulation. The first step is to formulate the spatial approximation of the equation using the finite element method, which results in a set of ordinary differential equations in time. This is known as a semidiscrete approximation. The second step is to solve the set of ordinary differential equations by employing either the finite element method again or the finite difference method.

It is customary to adopt the Galerkin approach of the finite element method for the spatial approximation and the Crank-Nicolson finite difference method for the temporal approximation [97,113].

It is worth noting that some problems, such as wave propagation, cannot be dealt with in this way as the spatial and temporal parts of the solution cannot be separated.

Some of the partial differential equations we will consider are those of a time dependent convection-diffusion type. Difficulties can arise for convection dominated flows as it is possible for results to be corrupted by spurious oscillations or wiggles. A relatively recent way of attempting to overcome this problem is by applying the streamline upwind/ Petrov-Galerkin formulation of Brooks and Hughes [11], taking care that over-damping of the solution does not occur.

3.6 Summary

In this chapter a brief explanation has been given of the numerical methods currently used in Newtonian and generalised Newtonian fluid mechanics, expressing a preference for the finite element method. Included in the text is a brief description of triangular and quadrilateral elements and the Galerkin weighted residual method, which have been selected for use in the development of the finite element method in this thesis. An overview is given of three different finite element formulations that are applied to fluid mechanics problems for steady flows, namely Stream Function-Vorticity, Velocity-Pressure and Penalty formulations. A brief explanation of the standard method used for time dependent steady flows is also given and mention made of upwinding for advection dominated flows.

The formulation which we have found has the most widespread usage is the U,V,P model and it is the formulation which will be used to solve the fluid mechanics problems presented in this thesis. For the time dependent steady flow problems we will use the Galerkin/Crank-Nicolson semi-discrete formulation and also the streamline upwind/ Petrov-Galerkin method for the advection dominated flow problems.

CHAPTER 4

NUMERICAL MODELLING OF DRIVEN LAMINAR FLOW IN A SQUARE CAVITY

4.1 Introduction

This particular problem is one of the standard test geometries. It was thought better to start with a simple two-dimensional problem to test the validity of the numerical model used to overcome any problems that might arise in developing the finite element method, and to check the accuracy of the results obtained, before proceeding with the more complicated three dimensional axisymmetric problem. Much work has already been done on the cavity problem by other researchers and many have developed finite element programs and have obtained results which could be used as a comparison (e.g. Cliffe et al [15]).

This problem is also known as a plate moving over a slot. The cavity enclosing the fluid has sides of unit length with three sides being stationary and the top moving from left to right with unit velocity, see fig. 4.1. There are singularities in velocities at the top corners. In addition to the given boundary conditions for the U and V velocities we specify the pressure to be zero at one point in the flow domain or on the boundary to overcome pressure arbitrariness.

In this chapter the equations associated with the flows due to a plate moving over a slot under both primary and secondary steady flow conditions are developed and the numerical methods used to solve them are outlined. Primary flows occur when Reynolds number

approaches zero, whereas secondary flow occurs at higher Reynolds numbers. Also in this chapter a time dependent dispersive mixing problem involving the same geometry is considered.

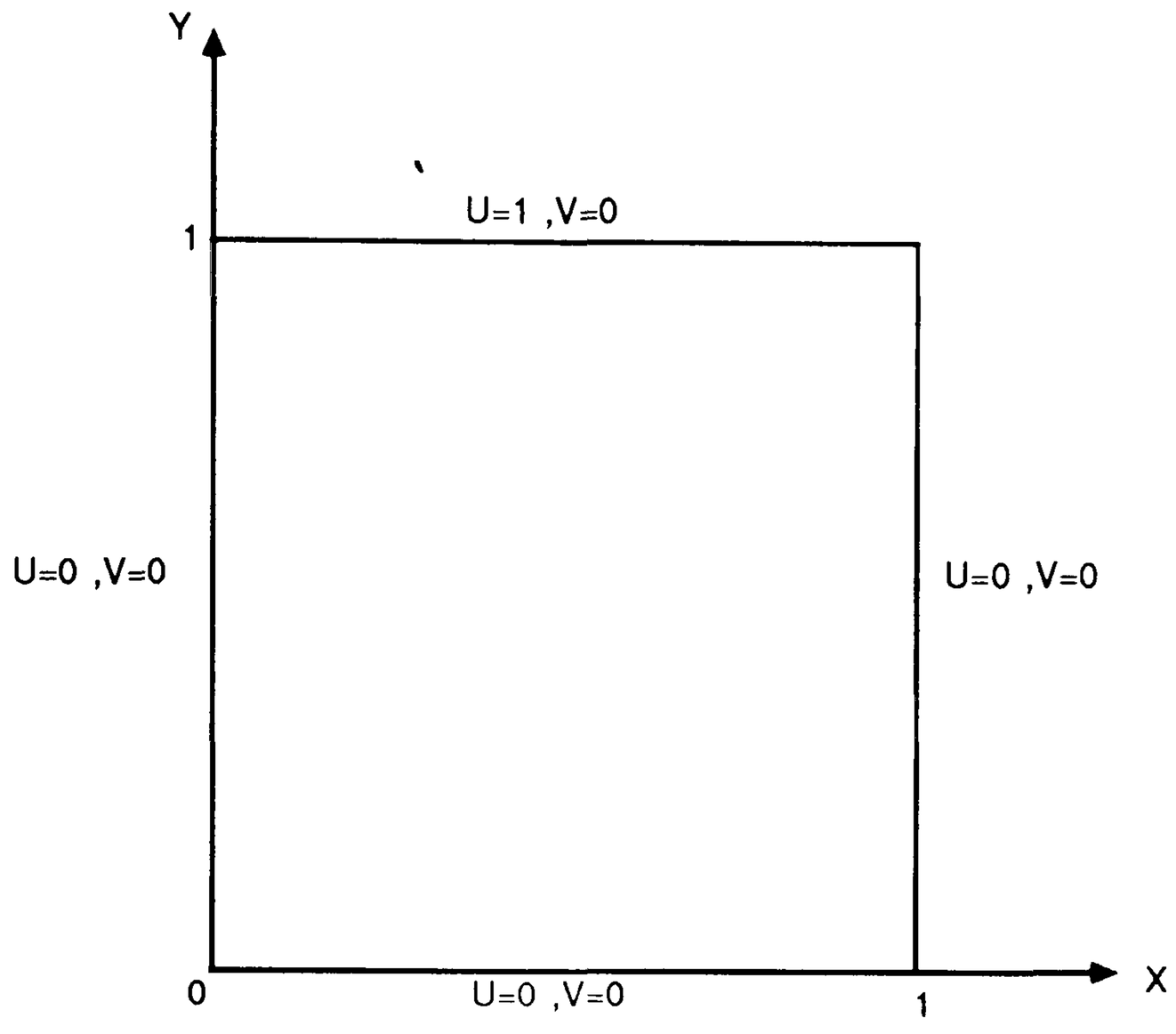


Figure 4.1 Diagram showing slot geometry including boundary conditions.

4.2 Equations for Steady Flow

The partial differential equations which describe the flows induced by a plate moving over a slot are:

i) the stress equations of motion

ii) the continuity equation

iii) the rheological equations of state.

4.2.1 The Stress Equations Of Motion

These equations are also known as the Cauchy Equations and they arise from the application of Newton's laws of motion to a moving continuum and the local expression of the principle of conservation of angular momentum. The equations describe the balance between the inertial forces, the pressures and the stresses due to the liquid itself (the principle of balance of linear momentum). These equations are the same for most materials, Newtonian or non-Newtonian. The equations are represented in terms of the stress in the x and y directions. We consider only steady (i.e. $\partial/\partial t=0$), isothermal flows with the density of the fluid taken as a constant. The equations of motion in standard tensor notation now reduce to

$$\rho[\, UU_{,x} + VU_{,y} \,] = -P_{,x} + T_{xx,x} + T_{xy,y} \qquad (4.1)$$

$$\rho[\, UV_{,x} + VV_{,y} \,] = -P_{,y} - \rho g + T_{xy,x} + T_{yy,y} \qquad (4.2)$$

where T_{xx}, T_{xy}, T_{yy} are physical components of the extra stress

tensor due to the structure of the fluid,

U,V are the velocities in the x and y directions respectively,

P is an arbitrary isotropic pressure,

ρ is the density of the fluid,

g is the gravitational constant,

and , denotes partial differentiation.

4.2.2 Continuity Equation

From the principle of conservation of mass for an incompressible fluid in a 2-dimensional plane flow we have, for constant density

$$U_{,x} + V_{,y} = 0 \quad . \qquad (4.3)$$

4.2.3 Rheological Equations Of State

These are also known as the constitutive equations and they relate stresses and strains in liquid deformation. In our case we take a purely viscous liquid.

$$T_{ik} = 2\eta(q)E_{ik} \quad , \tag{4.4}$$

where E_{ik} is the first rate of strain tensor representing velocity gradients with distance. There are four non-zero components of the rate of strain (and consequently stress tensor) and they are given by

$$E_{xx} = U_{,x} \quad , \quad E_{yy} = V_{,y} \quad ,$$

$$E_{xy} = E_{yx} = (1/2)(U_{,y} + V_{,x}) \quad , \tag{4.5}$$

with q being the representative deformation rate for the flow given by

$$q = \sqrt{[\, 2(\, E_{xx}^2 \; + \; E_{yy}^2 \; + \; 2(E_{xy})^2 \,)]} \quad . \tag{4.6}$$

One popular choice for the precise form of $\eta(q)$ in equation (4.4) is the Cross fluid model which represents shear-thinning behaviour (fig. 4.2) by

$$\eta(q) = \eta_\infty + \frac{\eta_0 - \eta_\infty}{1 + (\lambda q)^{1-n}} \quad , \tag{4.7}$$

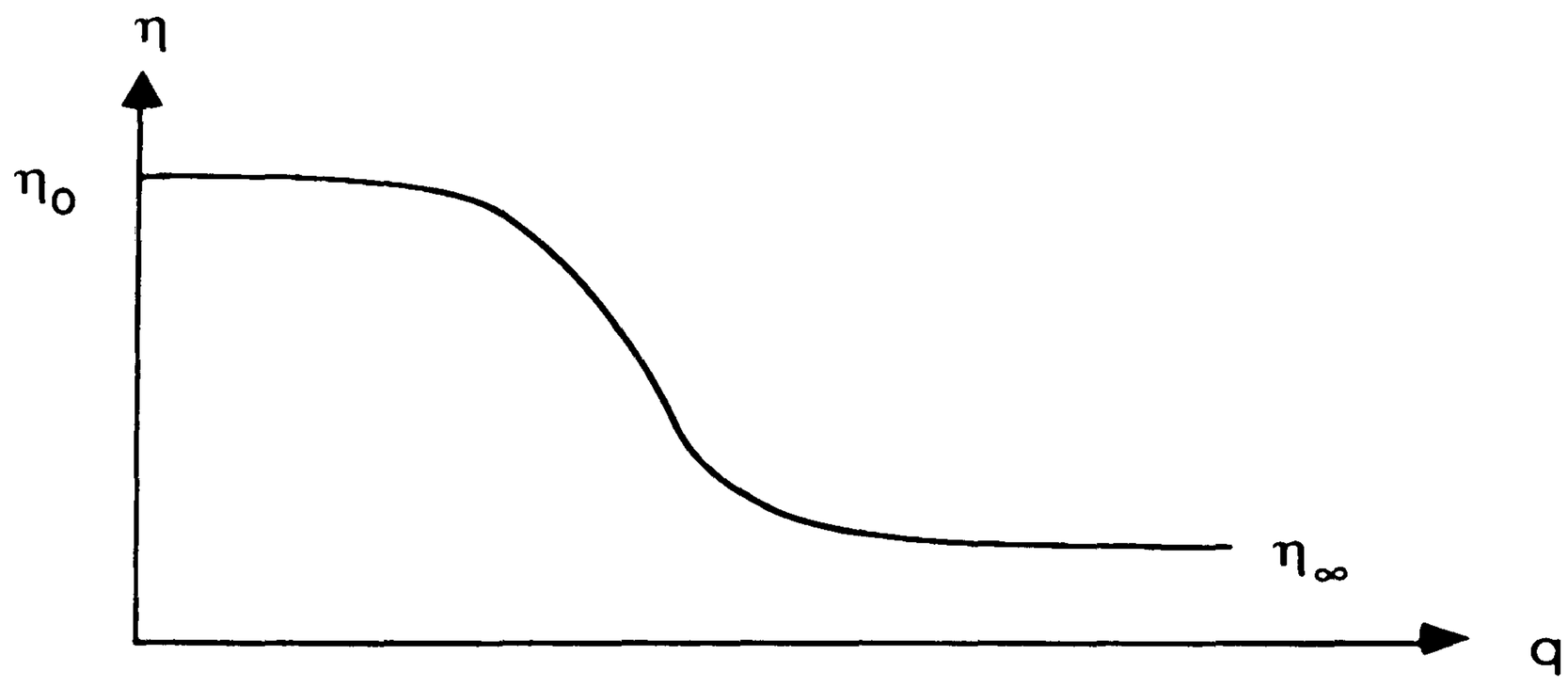


Figure 4.2 Graph showing Shear-Thinning behaviour.

where we take $\eta_0 = 1$ and is known as the zero shear viscosity,

$\eta_\infty = 0.1$ and is known as the infinite shear viscosity,

$\lambda=1$ and $n=0.5$.

When $n=1$, $\eta(q)$ reduces to a constant η_0, the Newtonian case.

4.3 Equations In Non-Dimensional Form

Writing the equations in a dimensionless form facilitates generalisation to embody a large range of problems. Non-dimensionalising in the usual way [42] we obtain the following equations

$$U'U',_{x'} + V'U',_{y'} = -P',_{x'} + 1/Re(T'_{xx},_{x'} + T'_{xy},_{y'}) \qquad (4.8)$$

and

$$U'V',_{x'} + V'V',_{y'} = -P',_{y'} - 1/(Fr)^2 + 1/Re(T'_{xy},_{x'} + T'_{yy},_{y'}) \qquad (4.9)$$

where $\rho=1$ and ' denotes a non-dimensionalised quantity. The continuity equation can now be written as

$$U',_{x'} + V',_{y'} = 0 \qquad . \qquad (4.10)$$

we have

$Re = \rho U_o l/\eta_o$, the Reynolds number,

and

$Fr = U_o/\sqrt{(gl)}$, the Froude number,

l is the characteristic length (= 1 for the slot),

Uo is the characteristic velocity (= 1 for the slot),

and gravity acts in the negative y' direction.

For convenience of presentation, the ' is omitted from the above equations for the rest of this chapter.

4.4 The U,V,P Finite Element Formulation

Initially we will develop the formulation for a general 2-D problem, followed by its application to a specific flow problem in section 4.9.

Conceptually (4.8),(4.9) and (4.10) present no major difficulties when implementing the F.E.M. for both spatial discretisation and variable definition. However, in this case the three equations must be satisfied simultaneously. If the equations were uncoupled then each could be solved independently and the corresponding variable evaluated. However, since strong coupling exists, a suitable solution algorithm is developed where the three equations are satisfied simultaneously.

The particular F.E.M. which was chosen to solve these partial differential equations was the U,V,P approach making use of a weak formulation of the Galerkin weighted residual technique. There are two schools of thought as to how the weak formulation should be applied to the equations:

(i) In scheme 1 the order of the pressure term is not reduced in the integral formulation of the equations of momentum and therefore P does not come into the boundary line integral. The overall system matrix is unsymmetric, non-positive definite. This scheme is adopted by Hughes and Taylor [42].

(ii) In scheme 2 the order of the pressure term is reduced in the integral formulation of the equations of momentum, giving natural boundary conditions in which P is specified. The overall system matrix is also unsymmetric and non-positive definite. This scheme is adopted by Crochet et al [19]. If the convective terms in the equations of momentum are neglected the overall system matrix produced is symmetric, as shown by Reddy [97].

There is no apparent advantage in choosing one scheme over the other, and in fact both were implemented for comparative purposes.

4.5 Scheme 1: U,V,P Finite Element Approach

In general the possible boundary conditions for a problem of this type fall into two categories :

(a) essential (Dirichlet) :

$$U = Ua \ , \quad V = Va \qquad \text{defined on } \Gamma_1 \tag{4.11}$$

$P = Pa$ at one point in the domain, and

(b) natural (surface forces) :

$$\tau_x = T_{xx}.n_x + T_{yx}.n_y$$

$$\text{defined on } \Gamma_2 \tag{4.12}$$

$$\tau_y = T_{yx}.n_x + T_{yy}.n_y \ ,$$

where n_x, n_y represent the outward normal components in the x and y directions respectively. Also

$$\Gamma = \Gamma_1 + \Gamma_2 \ ,$$

where Γ is the whole of the boundary of the domain (fig. 4.3 in text).

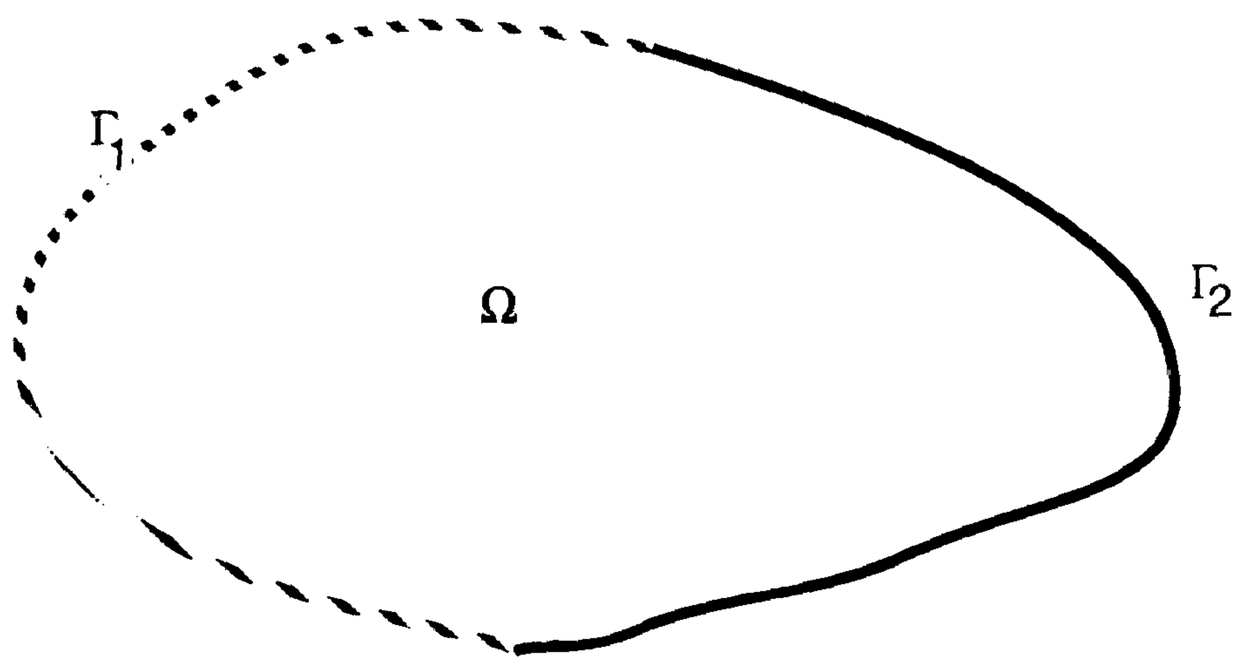


Figure 4.3 General region.

Note that P is only specified as an essential condition at one point (or node) which is usually taken to be inside the domain.

The Galerkin weighted residual method, as outlined in chapter 3 section 3 is applied to the equations of momentum and continuity, with U,V and P satisfing the essential boundary conditions. Taking S and N as the weighting functions, S being associated with U and V, and N associated with P, we have

$$< -P_{,x} + (1/Re)(T_{xx,x} + T_{yx,y}) - (UU_{,x} + VU_{,y}) ; S > = 0 \quad ,$$

$$< U_{,x} + V_{,y} ; N > = 0 \quad , \tag{4.13}$$

$$< -P_{,y} + (1/Re)(T_{yx,x} + T_{yy,y}) - 1/(Fr)^2 - (UV_{,x} + VV_{,y}) ; S > = 0 \quad ,$$

where $< ; >$ stands for the scalar product over Γ i.e.

$$< U ; V > = \iint UV \, dxdy \, .$$

We now require the weak formulation of the problem and this is obtained by applying the divergence theorem to the stress parts of the equations of momentum (4.13) giving,

$$< (1/Re)(T_{xx,x} + T_{yx,y}) ; S> = -< (1/Re)T_{xx} ; S_{,x}> - < (1/Re)T_{yx} ; S_{,y}> + \int_{\Gamma_2} (1/Re)\tau_x . S \, ds \tag{4.14}$$

$$< (1/Re)(T_{yx,x} + T_{yy,y}) ; S> = -< (1/Re)T_{yx} ; S_{,x}> - < (1/Re)T_{yy} ; S_{,y}> + \int_{\Gamma_2} (1/Re)\tau_y . S \, ds \tag{4.15}$$

We now substitute (4.14) and (4.15) into (4.13) to obtain the weak formulation of the problem. This is done to lower the order of the stress terms and relax the strong continuity requirements of (4.13). We obtain

$$<(1/Re)T_{xx} ; S_{,x}> + <P_{,x} ; S> + <(1/Re)T_{yx} ; S_{,y}> + <(UU_{,x} + VU_{,y}) ; S> = \int_{\Gamma_2} (1/Re)\tau_x . S \, ds$$

$$< U_{,x} + V_{,y} ; N > = 0 \tag{4.16}$$

$$<(1/Re)T_{yx} ; S_{,x}> + <P_{,y} ; S> + <(1/Re)T_{yy} ; S_{,y}> + <(UV_{,x} + VV_{,y}) ; S> = \int_{\Gamma_2} (1/Re)\tau_y . S \, ds + <1/Fr^2 ; S>$$

Now approximating U,V and P by $\tilde{U},\tilde{V}$ and $\tilde{P}$ where

$$\tilde{U} = \sum_{i=1}^{m} U_i.S_i \quad ; \quad \tilde{V} = \sum_{i=1}^{m} V_i.S_i \quad ; \quad \tilde{P} = \sum_{l=1}^{b} P_l.N_l \tag{4.17}$$

and S_i and N_l are shape functions, equations (4.16) can now be written as follows;

$$<2\eta(\tilde{q})(1/Re)\tilde{U}_{,x} ; S_{i,x} > + < \eta(\tilde{q})(1/Re)(\tilde{U}_{,y} + \tilde{V}_{,x}) ; S_{i,y} > + < \tilde{P}_{,x} ; S_i >$$
$$+ < \tilde{U}\tilde{U}_{,x} + \tilde{V}\tilde{U}_{,y} ; S_i > = \int_{\Gamma_2} (1/Re)\tilde{\tau}_x.S_i \, ds$$

$$< \tilde{U}_{,x} + \tilde{V}_{,y} ; N_l > = 0 \tag{4.18}$$

$$< \eta(\tilde{q})(1/Re)(\tilde{U}_{,y} + \tilde{V}_{,x}) ; S_{i,x} > + <2\eta(\tilde{q})(1/Re)\tilde{V}_{,y} ; S_{i,y} > + < \tilde{P}_{,y} ; S_i >$$
$$+ < \tilde{U}\tilde{V}_{,x} + \tilde{V}\tilde{V}_{,y} ; S_i > = \int_{\Gamma_2} (1/Re)\tilde{\tau}_y.S_i \, ds + (1/Fr^2) < 1 ; S_i >$$

where 1<=i<=m and 1<=l<=b. The values of m and b are dependent on the choice of element, so that for a triangular element we have m = 6 and b = 3. Also S and N must satisfy (4.16) for all values of i and l. For non-homogeneous Dirichlet boundary conditions on U and V, two functions U" and V" are defined which satisfy the essential boundary conditions, we have the approximation

$$\tilde{U} = U'' + \sum_{i=1}^{m} U_i.S_i \quad ; \quad \tilde{V} = V'' + \sum_{i=1}^{m} V_i.S_i \tag{4.19}$$

when imposing an essential Dirichlet condition on a specific node ,say j ,and the Galerkin equation for U is replaced by U_j = Ua. The above equations for a particular element can be written in matrix component form, as;

$$\mathbf{A1}_{ij} = <2\eta(\tilde{q})(1/Re)S_{j,x} ; S_{i,x} > + < \eta(\tilde{q})(1/Re)S_{j,y} ; S_{i,y} > ,$$

$$\mathbf{A2}_{ij} = < \eta(\tilde{q})(1/Re)S_{j,x} ; S_{i,x} > + <2\eta(\tilde{q})(1/Re)S_{j,y} ; S_{i,y} > ,$$

$$\mathbf{A3}_{ij} = < \eta(\tilde{q})(1/Re)S_{j,x} ; S_{i,y} > ,$$

$$\mathbf{A4}_{il} = < N_{l,x} ; S_i > ,$$

$$\mathbf{A5}_{il} = < N_{l,y} ; S_i > ,$$

$$\mathbf{A6}_{lj} = < S_{j,x} ; N_l > ,$$

$$\mathbf{A7}_{lj} = < S_{j,y} ; N_l > ,$$

$$\mathbf{A8}_{ijk} = < S_k.S_{j,x} ; S_i > ,$$

$$\mathbf{A9}_{ijk} = < S_k.S_{j,y} ; S_i > ,$$

and the vector components are :

$$X_i = \int_{\Gamma_2} \tilde{\tau}_x.S_i \, ds ,$$

$$Y_i = (1/Fr^2)< 1 ; S_i > + \int_{\Gamma_2} \tilde{\tau}_y.S_i \, ds ,$$

where $1<=i,j,k<=m$, $1<=l<=b$ and $\tilde{q}$ is obtained by substituting $\tilde{U},\tilde{V}$ for U,V in equations (4.6). We thus obtain the following algebraic system for a general element using suffix notation:

$$\mathbf{A1}_{ij}U_j + \mathbf{A3}_{ij}V_j + \mathbf{A4}_{in}P_n + (\mathbf{A8}_{ijk}\ U_j + \mathbf{A9}_{ijk}\ V_j)U_k = X_i \quad ,$$

$$\mathbf{A3}_{ji}U_j + \mathbf{A2}_{ij}V_j + \mathbf{A5}_{in}P_n + (\mathbf{A8}_{ijk}\ U_j + \mathbf{A9}_{ijk}\ V_j)V_k = Y_i \quad , \qquad (4.20)$$

$$\mathbf{A6}_{lj}U_j + \mathbf{A7}_{lj}V_j = 0 \quad .$$

This algebraic system is produced for every element in the mesh and is added into an overall stiffness matrix. The contributions for a particular node are from the element matrices directly connected to that node. These contributions are then summed and placed in the correct position in the system matrix. In this way the overall matrix is built up.

It is worth noting at this stage that the system of algebraic equations is non-linear with non-linearities arising from the convective terms in the equations of momentum and also from the nature of the fluid being considered if it is non-Newtonian.

To solve this non-linear problem either Newton's method or some other linearisation is used. We chose initially to linearise the problem by calculating the viscosity $\eta(\tilde{q})$ from (4.6) using old nodal values of velocity(i.e. previous iteration), beginning with an initial guess for these values. The same idea was used to linearise the convective term, setting a

part of the expression to be calculated at the old nodal velocity values. This form of linearisation was chosen because it is a simlpe technique which works well. The linearisation of the convective terms was similar to that employed by Hughes and Taylor [42]. Crochet et al [19] suggests using Newton's method for the convective terms and a similar linearisation procedure as ours for the terms involving shear viscosity, as an improvement to our algorithm we implemented Newton's method and a comparison of both methods is given later in the chapter.

It is necessary to impose a convergence criterion on the problem. This tolerance limit depends on how accurate a solution is required. We adopt a criterion of the form

$$|U_i^n - U_i^{n-1}| < e \quad , \quad |V_i^n - V_i^{n-1}| < e \quad , \quad |P_i^n - P_i^{n-1}| < e \quad , \tag{4.21}$$

n being the iteration number and e the tolerance limit. If the tolerance limit has not been achieved the current values of U and V are set to be a half of the newly calculated values and a half of the old values from the previous iteration, so that

$$\bar{U}^n = \frac{U^n + U^{n-1}}{2} \quad , \quad \bar{V}^n = \frac{V^n + V^{n-1}}{2} \quad . \tag{4.22}$$

Due to the fact that there is no pressure term in the pressure equation (i.e. the continuity equation) zeros appear on the leading diagonal of the overall system matrix, making it non-positive definite. Therefore equation solvers that employ some form of pivoting strategy have to be used. These solvers will introduce values onto the leading diagonal as the elimination of pivotal rows and columns from the matrix proceeds, making it possible to solve the system of equations. We have used the frontal elimination method, which is mentioned later in section 4.6.1.1. To obtain a solution at a specific Reynolds number it is customary to first obtain the solution at a low Reynolds number (e.g. Re =1) and then to increase the Reynolds number by reasonable sized steps, applying the iterative scheme to obtain the solution at each step, until the required Reynolds number is reached.

Until now we have not specified the kind of elements that are appropriate for our problem. We already know that the shape functions S for approximating the velocities have to have a higher order than those for the pressures. The most commonly used elements are triangles and rectangles, both of which are used in this thesis (fig. 4.4a and fig.4.4b).

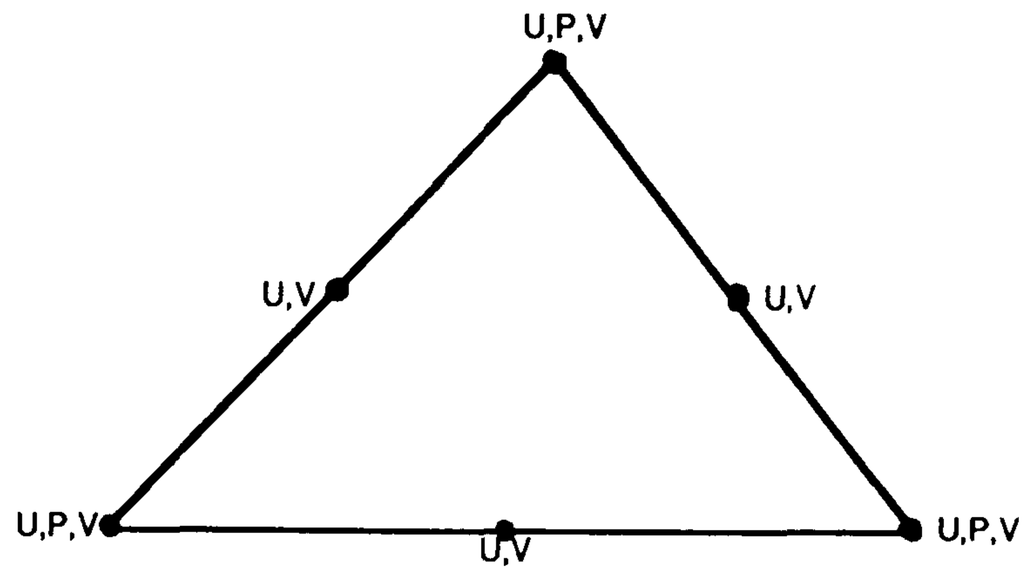


Figure 4.4a A triangular element with 15 degrees of freedom

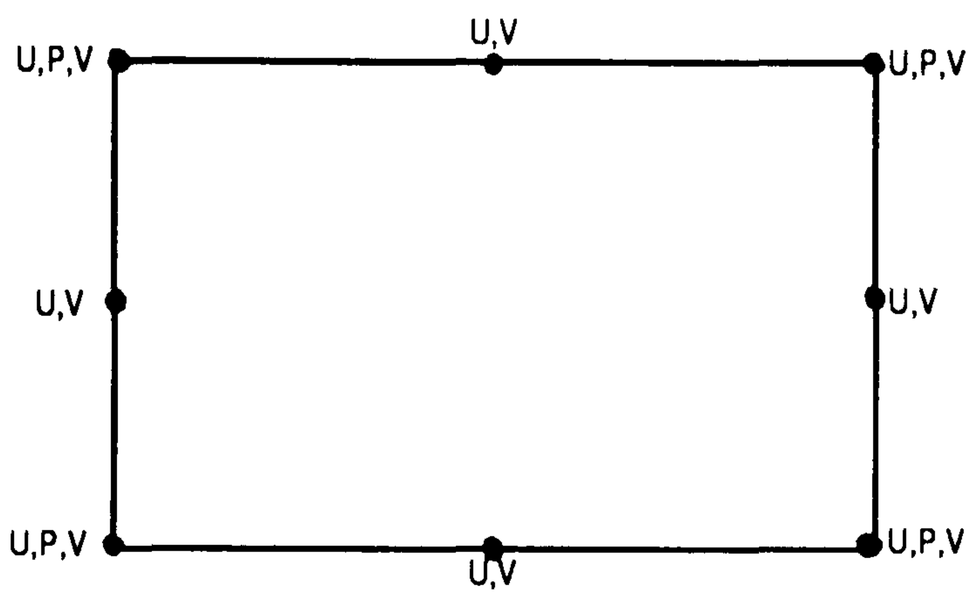


Figure 4.4b A quadrilateral element with 20 degrees of freedom

In keeping with the usual practice (Crochet, Davies and Walters [19]), the triangular and quadrilateral elements have pressures defined only at the vertices of the element, whilst the velocities are defined at the vertices and the mid-side nodes. For the triangular element the approximating function for pressure is a bi-linear polynomial containing three linearly independent terms and the velocities are specified by a bi-quadratic interpolation function with six coeffiecients. Whereas for the quadrilateral element the approximating function for pressure is again a bi-linear polynomial this time containing four linearly independent terms and the velocities are specified by a bi-quadratic second order polynomial with eight coeffiecients.

In flow problems such as these many workers state that the ratio of the number of discrete momentum equations to the number of continuity equations should be approximately two to one [19]. A drawback with both the triangular and serendipity elements is that this ratio is fairly high, that is, when two triangular elements (fig. 4.4a) are added into a large mesh of elements the ratio of new velocity nodes to pressure nodes introduced is eight to one whilst for the serendipity element (fig. 4.4b) when an extra element is added to a large mesh the ratio of velocities to pressures introduced is six to one, which is quite high compared to the optimal value of two to one. This may cause problems as the continuity constraint will be only weakly satisfied near corners or singularities.

4.6 The Finite Element Program And Associated Programs

A suite of programs were developed to solve the two-dimensional slot problem. For this problem the following programs were written to support the main program:

i) a finite element program (main program)

ii) a mesh generator program

iii) a mesh refinement program

iv) a stream function program (including contour plotting routine)

v) a Cuthill-Mckee program.

Below is a brief explanation of each of these programs:

4.6.1 The Finite Element Program

This program can be subdivided into three parts:

a) a main calling program

b) a frontal elimination solver

c) a element matrix assembler.

A number of subroutines from the Numerical Analysis group (NAg) finite element package [34] were incorporated into the code.

4.6.1.1 The Frontal Elimination Solver

The frontal method was originally devised by Irons [53] for the solution of symmetric sparse matrices but since then has undergone many revisions by others to accommodate unsymmetric matrices and various pivotal strategies. There are two frontal solvers that are used for our type of application and both of which are for the solution of unsymmetric matrices.

(i) The first, written by Hughes and Taylor [42] performs Gaussian elimination with only diagonal pivoting.

(ii) The second routine (Hood [41]) performs Gaussian elimination with full pivoting.

Reid [98] has also suggested the use of a multi-frontal solver. The basic concept behind this solver is to have a number of fronts which start simultaneously from different parts of the boundary, eventually all the fronts are concatenated so that the elimination process can be completed. The potential advantage of this algorithm is that it is likely to work when a standard frontal solver will fail and is applicable to parallel computation, and this type of solver is explained in greater detail in chapter 6.

For our purposes the first of the two elimination routines is used, because the computation time involved is much less and full pivoting is not required in this class of problem since the largest values are usually on the diagonal. Even though initially there are a number of zeros on the diagonal these are filled in as the elimination procedure proceeds. The Hood solver [41] was tried but comparatively there was no significant enhancement in the results obtained and a considerable increase in computation time was noticed.

The main aim of the Hughes and Taylor frontal method [42] is to eliminate variables from the system matrix as soon as possible, allowing for the reduction of the matrix so that Gaussian elimination is performed on matrices far smaller than the complete overall system matrix. This makes a large saving in computation time and memory storage, since the complete system matrix is never actually fully assembled. The algorithm is split into three main parts:

a) the element assembly

b) the elimination process

c) the back substitution process.

A fuller description is given in chapter 6, and for further details see also Hughes and Taylor [42].

4.6.1.2 **Assembly of the Element Matrix**

Implemented in this routine is the equation set (4.20) which produces for a particular element its element stiffness matrix. The element stiffness matrix is then added into its correct position in the frontal matrix as directed by a header vector.

4.6.2 **Mesh Generator Program**

This generates a data file containing all necessary information relating to the mesh of triangular elements to cover the domain of a specified geometry, i.e. the nodal coordinates, element topology e.t.c. A sensible ordering of the elements is essential so that the size of the frontwidth is minimised.

The choice of mesh is largely intuitive, even though it is useful to have an understanding of what the solution should look like. There are a few guidelines which are worth noting when deciding on a mesh for a particular problem, as the choice of mesh can affect the accuracy of the solution obtained.

i) The mesh should not contradict with the physical or mathematical symmetry which is manifested in the problem, if any.

ii) The mesh should be graded or refined near parts of the geometry which are of greater interest, since at some places the flow is more complex than elsewhere. Mesh refinement should be used for specific parts of the geometry when there are:

a) sharp changes in geometry,

b) abrupt changes in boundary conditions on particular boundary walls,

c) "sudden" changes in the medium,

d) "sudden" changes in the solution.

4.6.3 **Mesh Refinement Program**

This program performs an overall mesh refinement of a mesh that has already been created. Initially a solution can be obtained using the original mesh which is believed to be adequate for the problem. The mesh is then refined until the differences in the solutions agree to a predefined tolerance.

4.6.4 **Stream Function Program**

We require at this stage to be able to calculate the stream function values at the nodes so that plots of streamlines can be drawn, enabling comparison with the work of others. From the finite element program we obtain the solution in terms of U and V (and also P). The relationship between the stream function ϕ and the velocities is:

$$U = \phi_{,y} \tag{4.23}$$

$$V = -\phi_{,x} \quad . \tag{4.24}$$

It can be seen from the above equations that ϕ satisfies the Poisson equation

$$\phi_{,xx} + \phi_{,yy} = U_{,y} - V_{,x} \quad , \tag{4.25}$$

with $\phi = 0$ on Γ .

To obtain the stream function we solve (4.25) by employing the Rayleigh-Ritz formulation of the finite element method and using a Choleski symmetric band matrix solver. Added to the end of the program is a simple contour plotting routine.

4.6.5 The Cuthill-Mckee Program

This program applies the Cuthill-Mckee algorithm [21] to the data to be used by the stream function program, for a sparse structured symmetric matrix. The main aim of the program is to minimise the half-bandwidth of the overall system matrix by reordering the nodes, saving on memory storage and computation time.

4.7 Scheme 2: U,V,P Finite Element Approach

In this section a brief outline is given of this method, which was mentioned at the beginning of section 4.4. Scheme 2 is very similar to the first scheme which has been developed in section 4.5 and 4.6, and therefore only the main differences will be explained here.

The scheme is also based on the Galerkin weak formulation of the U,V,P approach finite element method. In this scheme the order of the pressure term is also reduced in the integral formulation of the momentum equations and gives a natural boundary condition in which pressure is specified.

This weak formulation is obtained by applying the divergence theorem to the stress and pressure parts of the equations of momentum (4.13) to give

$$< (1/Re)(T_{xx,x} + T_{yx,y}) - P_{,x} ; S> = - < -P + (1/Re)T_{xx} ; S_{,x}> - <(1/Re)T_{yx} ; S_{,y}> + \int_{\Gamma_2} \tau_x . S \, ds \qquad (4.26)$$

$$< -P_{,y} + (1/Re)(T_{xy,x} + T_{yy,y}) ; S> = - <(1/Re)T_{xy} ; S_{,x}> - <-P + (1/Re)T_{yy} ; S_{,y}> + \int_{\Gamma_2} \tau_y . S \, ds \ , \qquad (4.27)$$

where

$$\tau_x = (-P + (1/Re)T_{xx}).n_x + (1/Re)T_{yx}.n_y \quad \text{on } \Gamma_2 \tag{4.28}$$

$$\tau_y = (1/Re)T_{yx}.n_x + (-P + (1/Re)T_{yy}).n_y \quad .$$

To continue with this scheme the above equations are substituted back into equations (4.13) and the formulation is proceeded with much in the same way as in section 4.5. On reaching the matrix component form it is noticed four of the components are different for scheme 2 to those described in scheme 1 (4.20), and the differences are:

$$\mathbf{A4}_{il} = < N_l ; S_{i,x} > \quad ; \quad \mathbf{A5}_{il} = < N_l ; S_{i,y} >$$

with $\mathbf{A6}_{lj}$ replaced by $\mathbf{A4}_{jl}$ and $\mathbf{A7}_{lj}$ replaced by $\mathbf{A5}_{jl}$. These alterations are simple to make in the finite element program, and as a final change the pressure now appears in the boundary integral in Xi on the right hand side.

Dirichlet conditions for P are a natural boundary condition for this formulation and P has also to be specified at least at one point on the boundary in order for determinacy. This scheme was then used to solve the slot problem defined in section (4.1), so that a comparison could be made of both schemes. As the pressure is not always known on the boundary it was envisaged that this might cause some difficulty in calculating the natural boundary condition , but this difficulty did not arise in our problem since Dirichlet boundary conditions were specified for U and V all the way around the boundary and it was thus unnecessary to specify values for pressure in the line integrals of the U and V equations on the boundary. When essential conditions are specified the U and V equations for the particular boundary points are discarded and replaced by the imposed boundary conditions. If there were a Neumann boundary condition specified on a boundary for either U or V, then it would have been necessary to specify a value for P which would be inserted into the line integral for that particular boundary, namely (4.15) see [19].

The results from this scheme applied to the problem of a plate moving over the a slot agreed almost identically with the results previously obtained for scheme 1. There were no real differences in the coding of either scheme, with the speed of both algorithms being equivalent and memory storage the same. This suggested that both schemes were equally as applicable and there was no real preference as to which one should be used.

4.8 Application of Newton's Method

The finite element programs we have written use a considerable amount of C.P.U. time. One way of reducing this time is to employ a method which takes a fewer number of iterations to converge on a solution. The method previously adopted has linear convergence whereas a method, such as Newton's method, has a rate of convergence which is usually quadratic.

Newton's Method is very well known and can be found in a number of texts e.g. [19]. Newton's method is applied to the matrix component form of the algebraic system of equations (4.20) and the new resulting equations are presented here as:

$$(\mathbf{A1}_{ij}+ (\mathbf{A8}_{ijk}+ \mathbf{A8}_{ikj})U_k^n + \mathbf{A9}_{ikj}V_k^n)\delta U_j+ (\mathbf{A3}_{ij}+\mathbf{A9}_{ijk}U_k^n)\delta V_j+ \mathbf{A4}_{ip}\delta P_p$$
$$= X_i- \mathbf{A1}_{ij}U_j^n- \mathbf{A3}_{ij}V_j^n- \mathbf{A4}_{ip}P_p^n- (\mathbf{A8}_{ijk}U_j^n+ \mathbf{A9}_{ijk}V_j^n)U_k^n$$

$$(\mathbf{A3}_{ji}+\mathbf{A8}_{ijk}V_k^n)\delta U_j+(\mathbf{A2}_{ij}+\mathbf{A8}_{ikj}U_k^n+\mathbf{A9}_{ijk}V_k^n+\mathbf{A9}_{ikj}V_k^n)\delta V_j+\mathbf{A5}_{ip}\delta P_p$$
$$=Y_i-\mathbf{A3}_{ji}U_j^n-\mathbf{A2}_{ij}V_j^n-\mathbf{A5}_{ip}P_p^n-(\mathbf{A8}_{ijk}U_j^n+\mathbf{A9}_{ijk}V_j^n)V_k^n \qquad (4.29)$$

$$\mathbf{A6}_{lj}\delta U_j+\mathbf{A7}_{lj}\delta V_j=-\mathbf{A6}_{lj}U_j^n-\mathbf{A7}_{lj}V_j^n \quad ,$$

where $\delta U_j, \delta V_j, \delta P_p$ represent the incremental difference of successive iterates i.e.

$$\delta U = U^{n+1} - U^n \quad , \quad \delta V = V^{n+1} - V^n \quad , \quad \delta P = P^{n+1} - P^n \quad . \qquad (4.30)$$

In solving equations (4.29) values for the incremental differences δU, δV and δP are returned and are subsequently used to obtain the updated values of U^{n+1}, V^{n+1} and P^{n+1} from the equations (4.30) in a re-ordered form, for example

$$U^{n+1} = U^n + \delta U \quad . \qquad (4.31)$$

The customary practice used to signify how well the iterative method is converging on a solution is indicated by the way in which the absolute value of the right hand side of the equations (4.29) approaches zero, stopping the scheme when the values for the right hand side are less than a preset convergence value.

To compare both methods the slot problem was again considered employing the finite element mesh fig. 4.5* and recording the number of iterations from each starting value to reach Re =100, taking four steps in Reynolds number to attain this value. It was checked that the frontwidth and convergence criterion used were the same for both methods and a table which presents the results is:

no. of iterations from	Reynolds number			
last Re value converged	1	10	50	100
linearisation method	8	3	6	6
Newton's method	3	2	2	3

From these results it is clearly seen that a worthwhile improvement in the number of iterations is possible when using Newton's method.

NOTE: *Certain figure numbers have a superscripted asterisk indicating that the respective figures are positioned at the end of the chapter. This continues throughout the remainder of the thesis.*

4.9 Results And Conclusions

In this section we discuss the results obtained for the slot problem including comparisons with the results obtained by other researchers for checking our method.

For this problem we defined the finite element mesh in a similar way to those presented in a report by Cliffe et al [15]. We considered two meshes, a fairly coarse mesh of 200 elements and 21x21 nodes, (fig. 4.5*), and a second more refined mesh with 800 elements and 41x41 nodes in the mesh, (fig. 4.6*).

(i) Newtonian Fluid Case

The streamline plots we have presented are those obtained using the refined 41x41 noded mesh. The three plots that have been incorporated in this section are at different Reynolds numbers, the first being at Re=1, (fig. 4.8*), the second at Re=400, (fig 4.10*) , and thirdly at Re=1000, (fig 4.12*) . The reason for choosing these three plots was to enable us to compare with the streamline plots produced by other researchers for meshes of similar refinement.

4.9.1 The Work Of Chien, Rising and Ottino [14]

In their report Chien et al present experimental and theoretical work on slot flows. Included in these is a similar geometry to ours with the width 11.0cm, and height 6.5 cm, and the plate moving from left to right with a velocity of 0.9 cm/s. Their streamline plots are shown at Re=0.6, (fig. 4.7a* and fig. 4.7b*). The theoretical results were obtained using a numerical scheme based on the finite element method for the same configuration as performed experimentally. In a text by Ottino [87] he presents experimentally produced streamlines for the slot problem, the geometry has the following dimensions; width 10.3 cm and height 6.2 cm and the Reynolds number is 1.7, see fig. 4.7c*. These streamline

plots at Re=0.6 and Re=1.7 are similar to our plots at Re=1(fig. 4.8*), the primary vortex being of the same shape and position in the flow domain , that is with the centre of the vortex being $3/4$ of the way up the cavity and symmetrical about the vertical centre line.

4.9.2 The Work Of Cliffe et al [15]

The geometry we use is the same as that used by Cliffe et al, and the numerical method used is the U,V,P finite element approach with meshes of varying degrees of refinement. Presented in fig. 4.9* is a streamline plot at Re=1, which was produced from a 57x57 noded mesh with top corner refinement, this is considerably more refined than ours. However their streamline plot for the same stream function values is almost identical to the one we have obtained (fig. 4.8*).

Presented also in Cliffe et al's report is a streamline plot at Re=400, (fig 4.11*), which can be compared with our fig 4.10*. The stream function values for the main vortex are the same in both plots, but our values for the secondary eddy are half those of Cliffe et al. The streamline plots are very similar although Cliffe et al's results were again obtained from a more refined mesh than the one we used. The centre of the main vortex has moved downwards and to the right as Re increases, while a secondary eddy has appeared in the bottom left corner circulating in the opposite direction to the main vortex.

In Cliffe et al's report mention is made that the maximum value of the stream function in the secondary eddy is double the strength of that without using refinement, which agrees with our results as our secondary eddy is half the strength of that obtained by Cliffe et al. Comparison was also made of U and V velocities along the vertical and horizontal centre lines both at Re=1 and Re=400, with our results which following all the trends of those presented by Cliffe et al, with a small percentage error due to the lack of corner refinement. It can be seen that our results are in excellent agreement with those of Cliffe et al.

4.9.3 The Work Of Mizukami [75]

Here we compare fig. 4.12* with results produced by Mizukami, who presents a streamline plot at Re=1000 ,(fig 4.13*), for the same geometry as that defined in fig. 4.1. In Mizukami's report the stream function-vorticity finite element method is used to solve the Navier-Stokes equations, and it can be seen from the plots that the primary vortex is being forced more into the top right corner with the centre of the vortex becoming more central in the flow domain. The secondary eddy in the bottom right corner has become stronger. Also another fairly weak eddy has appeared in the bottom left corner which circulates in the opposite direction to the main vortex. These plots are in very good agreement with each other even though the streamlines in these plots may not be the same values as Mizukami , as the values at which the contours have been plotted are not specified in Mizukami's report.

4.9.4 The Work Of Others

Bozeman and Dalton [9] considered the flow of a Newtonian fluid in rectangular cavities by solving various implicit finite difference approximations of the Navier-Stokes equations. In their paper the behaviour of the vortex centre and secondary vortices with increasing Reynolds number was exhibited. Initially for Re=0 the vortex centre was three-quarters of the way up and central in the cavity, and as Re is increased the centre of the vortex moves to the right and then slowly moves down and back towards the centre of the cavity, as Re gets large. This behaviour is consistent with the Batchelor model for large Reynolds numbers. The path which the vortex centre follows is similar in shape to a question mark. In our results the vortex centre was seen to behave in a similar manner to that exhibited by Bozeman and Dalton.

Nallasamy and Krishna Prasad [79] performed a numerical study of the flow in a square cavity at high Reynolds numbers, employing finite difference techniques. They also studied the behaviour of the vortex centre as a function of Reynolds number and agreed with the

results of Bozeman and Dalton. They also presented streamlines obtained at Re=30000, where the flow closely resembles inviscid flow with the centre of the primary vortex coinciding with the geometric centre of the cavity and the disappearance of the secondary eddies.

(ii) **Non-Newtonian Fluid**

We considered a non-Newtonian fluid, making use of the Cross model to simulate the properties of a shear thinning fluid. We present plots at Re=1 and Re=200, in figures 4.14 *and 4.15*. Comparison of Fig. 4.14* with Newtonian results at Re=1, (fig. 4.8*), shows that for the variable-viscosity fluid the main vortex has moved slightly up and to the right. This is due to the shear thinning nature of the fluid allowing the fluid to move more swiftly near the moving plate. This effect becomes more apparent when we consider the streamline plot for the variable viscosity fluid at Re=200, (fig 4.15*) , when compared with the Newtonian case at Re=200, (fig. 4.16*). The flow pattern of the non-Newtonian fluid resembles the flow pattern of a Newtonian fluid specified at a much higher Reynolds number, but the circulation is weaker, (fig 4.10*). The centre of the vortex for the variable-viscosity fluid is lower nearer the centre vertical line than for the Newtonian case, a trend much as expected.

4.10 Time Dependent Steady Flows - Application to Colour Band Mixing

In this section consideration is given to the theoretical investigation of colour band and dispersive mixing applied to mass transfer within the square cavity. The concentration of colour initially introduced in one area of the moving fluid and the subsequent concentration levels are calculated at various time intervals until a homogeneous mixture is obtained. This can be achieved by solving the non-dimensional form of the time-dependent concentration equation (4.32).

$$C_{,t} - D(C_{,xx} + C_{,yy}) + (UC_{,x} + VC_{,y}) = 0 \qquad (4.32)$$

where C signifies the concentration, D the diffusion coefficient given by

$$D = 1/(ReSc) \quad , \tag{4.33}$$

Re represents the Reynolds number, Sc the Schmidt number(taken to be 50) and U and V are the velocity components calculated from chapter 4. The equation is composed of three terms; the first term is the time-dependent part, the second term the diffusion contirbution and the third term the advection part. The boundary conditions are of a homogeneous Neumann type, whereas the initial conditions are shown in fig.4.17 below.

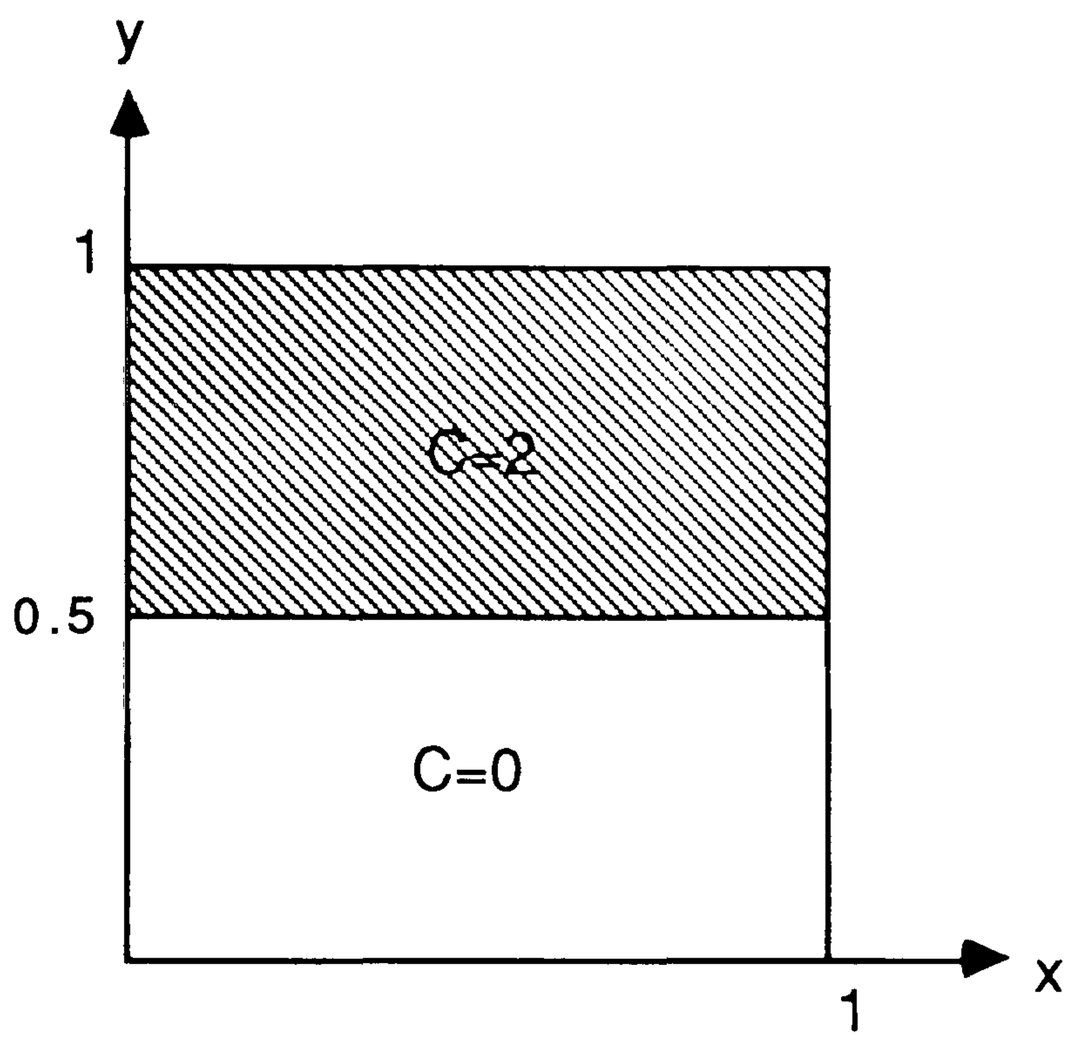


figure 4.17

The amount of concentration in the cavity is given by

$$\iint C \, dA = \text{constant.} \tag{4.34}$$

With the constant being determined from the initial concentration by

$$\int_0^1 \int_{0.5}^1 2 \, dxdy = 1 \tag{4.35}$$

The velocities at specific Reynolds numbers are known from results which were obtained using the U,V,P finite element program described earlier, and so the flow is considered to have already reached steady state before colour is added.

As the mixing takes place the overall concentration of colour in the region remains the same, that is equal to its initial value, and so can be used as a check on how well the solution is behaving.

The semidiscrete formulation is chosen as the method of solution, where the Galerkin finite element approach is employed for the spatial approximation and the Crank-Nicolson finite difference method is adopted for the temporal approximation. As the flow for high enough Reynolds number is advection dominated it is expected that the solution will be corrupted by spurious oscillations and so the introduction of some form of upwind approach will be required as with finite differences. To attempt to overcome the numerically induced wiggles the streamline upwind/ Petrov-Galerkin (SU/PG) method of Brooks and Hughes [11] will be implemented.

For the semidiscrete approach an approximation is required of the form

$$C(t,x,y) = \sum_{j=1}^{m} S_j(x,y)C_j(t) \quad , \tag{4.36}$$

where S(x,y) is the spatial term expressed as S and C(t) is the temporal term expressed as C throughout the rest of this formulation. The first stage of the formulation applies the Galerkin method to the spatial terms in the equation (4.32).

$$\iint (C_{,t} - D(C_{,xx} + C_{,yy}) + UC_{,x} + VC_{,y})S_i \, dxdy = 0 \quad . \tag{4.37}$$

Appling the divergence theorem to the second order terms we obtain

$$\iint D(C_{,xx} + C_{,yy})S \, dxdy = -\iint D(C_{,x}S_{,x} + C_{,y}S_{,y})dxdy + D\int SC_{,n} \, ds = 0 \quad . \tag{4.38}$$

Substituting (4.36) and (4.38) into (4.37) and adopting tensor product notation gives

$$C_{j,t}\langle S_j;S_i\rangle + C_j[\, D(\langle S_{j,x};S_{i,x}\rangle + \langle S_{j,y};S_{i,y}\rangle) + (U\langle S_{j,x};S_i\rangle + V\langle S_{j,y};S_i\rangle)] - D\int S_i S_{j,n} \, ds = 0 \quad , \tag{4.39}$$

which can be written in the matrix form

$$[A]\{C_{,t}\} + [B]\{C\} = 0 \quad , \tag{4.40}$$

where {} represents a column vector, [] represents a MxM matrix, the components of **A** are obtained from the tensor product term associated with the $C_{,t}$ term in equation (4.39), and the components of **B** are the tensor product terms that are inside the square brackets in (4.39) .

The next stage in the formulation is to apply the Crank-Nicolson finite difference approach to (4.40), which represents a set of first order differential equations in time. The approximation takes an average of the time derivative at two consecutive time steps by linear interpolation of the values of the variable at the two time steps and is given by,

$$\frac{\{C_{,t}\}^{n+1} + \{C_{,t}\}^{n}}{2} = \frac{\{C\}^{n+1} - \{C\}^{n}}{\delta t} \qquad (4.41)$$

with n signifing the iteration number. Substituting (4.40) into the above approximation and performing elementary matrix operations the matrix equation

$$-\delta t/2\ [\mathbf{B}]\{C\}^{n+1} - \delta t/2\ [\mathbf{B}]\{C\}^{n} = [\mathbf{A}]\{C\}^{n+1} - [\mathbf{A}]\{C\}^{n} \qquad (4.42)$$

is obtained. Rearranging the equation to place the subscript n+1 terms on one side of the equation and the n terms on the other gives

$$([\mathbf{A}] + \delta t/2[\mathbf{B}])\ \{C\}^{n+1} = ([\mathbf{A}\} - \delta t/2[\mathbf{B}])\ \{C\}^{n} \quad , \qquad (4.43)$$

where δt represents the time step.

The spatial approximation needs only to be worked out for the first value of time as it then remains constant throughout the calculation due to the velocities being steady-state velocities. The matrices obtained at the temporal stage are unsymmetric and so we apply a Gaussian reduction routine to the matrices for the first time step and after that it is merely a case of performing the back substitution at each successive time step, which is a large saving in computation time. The finite element mesh (fig.4.5*) composed of six noded triangular elements was used.

In testing the program we obtained satisfactory results for Re=1 (fig. 4.18*). The solution was beginning to diverge at Re=5 with oscillations observed in the values obtained for the overall concentration. It was surprising that the breakdown of the solution occurred for such a low Reynolds number, yet it lead us to assume that the reason for this breakdown lay in the flow being advection dominated. When spurious oscillations occur for advection dominated flows the customary practice is either to significantly refine the mesh or apply an upwinding technique.

The upwind method was adopted as it offered a solution to our problem without having to severely refine the mesh. For advection dominated flows the deficiency is connected with the numerical method not the actual physics of the problem. There is a similarity between the Galerkin method and central differences employed in the finite difference method in that there is a lack of diffusion present for a problem of this kind. To correct this deficiency an upwind method is applied which adds the "correct" quantity of artificial diffusion into the problem. If too much artificial diffusion is introduced over-damping of the solution can occur and so produce meaningless results.

There are a number of different classical upwind methods for multidimensional problems presented in papers, with some suggesting upwinding merely the advection term while others suggest adding an upwind contribution to the weighting functions throughout the problem producing a Petrov-Galerkin formulation, where the upwind pertubations employed are one-dimensional techniques which have been extended to multidimensional problems. These schemes suffer from either crosswind diffusion or overly diffuse solutions. The method we have adopted is the consistent streamline upwind/ Petrov-Galerkin method (SU/PG) of Brookes and Hughes [11] in which the "correct" amount of artificial diffusion is introduced into the problem and is only applied in the direction of the streamlines. This method overcomes the inherent diffculties associated with other classical upwind methods.

In the SU/PG method the shape functions used for the approximation functions and weighting functions are different, with the upwind pertubation only being added to the weighting function throughout the equation including the temporal term. Brookes and Hughes, in their paper, recommend using bi-linear quadrilateral elements in the finite element mesh. Up to this point we have only used triangles as elements so a simple program was written to transform the six noded triangular mesh data to a mesh of four noded quadrilateral elements.

For the SU/PG it is necessary to have discontinuous weighting functions of the form

$$S = S + P \tag{4.44}$$

where S is the continuous weighting function, the same as the test function, and P is the discontinuous streamline upwind pertubation. The form of P is given by

$$P = \frac{D\,(US_{,x} + VS_{,y})}{(U^2+V^2)} \quad , \tag{4.45}$$

where D is the scalar artificial diffusivity to be defined later in this section. Applying the SU/PG formulation to the equation (4.32) we obtain

$$\iint(C,t - D(C,xx + C,yy) + UC,x + VC,y)(S+P)dxdy = 0 \quad , \tag{4.46}$$

It is not necessary for P to be added to the diffusion term as second order derivatives are zero for bi-linear approximating functions, giving

$$\iint(C,t - D(C,xx+C,yy) + UC,x + VC,y)S\ dxdy + \iint C,tP\ dxdy + \iint(UC,x +VC,y)P\ dxdy = 0 \quad . \tag{4.47}$$

Here the first integral is identical to (4.37) which has been described earlier in this section and so attention is given to the integrals involving the upwind pertubation P. For the integral involving C,t we obtain

$$C_{j,t}\ \left(\frac{D}{(U^2+V^2)}\right)[\ U\langle S_j;S_{i,x}\rangle + V\langle S_j;S_{i,y}\rangle])$$

and the $C_{,j}$ integral takes the form

$$C_j \frac{(D}{(U^2+V^2)} [U^2\langle S_{j,x};S_{i,x}\rangle + UV\langle S_{j,x};S_{i,y}\rangle + VU\langle S_{j,y};S_{i,x}\rangle + V^2\langle S_{j,y};S_{i,y}\rangle]) .$$

These terms are then incorporated into (4.47) and the remaining stages of the formulation are the same as those earlier described for the non-upwinded case. The artificial diffusivity D for the transient case takes the form

$$D=(l_x U h_x + l_y V h_y)/\sqrt{15} \quad , \tag{4.48}$$

where $l_x = \mathrm{Coth}(\alpha_x) - \frac{1}{\alpha_x}$ and $l_y = \mathrm{Coth}(\alpha_y) - \frac{1}{\alpha_y}$

$$\alpha_x=\frac{Uh_x}{2D} \quad , \quad \alpha_y=\frac{Vh_y}{2D} \quad ,$$

h_x and h_y are element characteristic lengths, and U, V and D are evaluated at the element centre.

The original program was taken and the alterations required to incorporate the SU/PG formulation were carried out. The upgraded program managed to be successful in reaching Re=10, but on attempting to reach Re=100 the solution broke down again producing large oscillations in the value for the calculated concetration. This was very disappointing as we were confident that the upwinding scheme would have overcome the difficulty. On viewing plots of the flow leading up to the breakdown, it was noticed that the flow pattern in the upper portion of the flow domain was unstable, whereas the flow pattern in the lower portion remained fairly stable (fig 4.19*). This would suggest that the reason for the solution breaking down at such a low Reynolds number is due to the singularities at the top corners of the domain causing discontinuities to occur in the velocity derivatives near these points.

My supervisor Dr R.W.Williams decided to investigate this problem by using various upwind finite difference approaches. He also ran into the same difficulties that we had encountered and so it was decided not to give any further effort to the analysis of this difficulty in view of the more pressing and interesting work ahead. This was not found to

be such a difficulty in the rotating flows to be considered in chapter 5.

In a recent series of papers on new upwind methods by Hughes et al [43,44,45,46] they state that the SU/PG method works well for problems which have smooth solutions, but when sharp internal or boundary layers are present oscillations will normally appear in the vicinity of these sharp layers. To overcome this difficulty they suggest adding a discontinuity-capturing term to the SU/PG formulation. This would seem to indicate a way forward but for the reasons previously mentioned we do not intend to continue any further with this particular problem.

Due to the problems encountered with dispersive mixing for the cavity flow problem for a Newtonian fluid it was obvious that a similar problem would arise in the case of the Non-Newtonian fluid. For this reason no consideration was given to the Non-Newtonian case. It is worth noting that the procedure for formulating the problem for the Non-Newtonian case would be identical to that for the Newtonian case except for the use of different velocity values.

4.11 **Summary**

To conclude we have successfully implemented the primitive variable finite element formulation for a two dimensional test problem for both Newtonian and variable viscosity fluids. The results obtained compare well with the work of others. We have also implemented a semidiscrete formulation incorporating the streamline upwind Petrov/Galerkin approach for the solution of the time dependent advection/diffusion equation. For low Reynolds numbers it was possible to obtain sensible solutions but at higher Reynolds numbers the results became very unstable, we believe the reason for this was due to the singularities in the U velocity at the top corners of the slot.

The solution of the 2-D slot problem was very useful as an introduction to the finite element method in the context of numerical simulation of fluid flow problems.

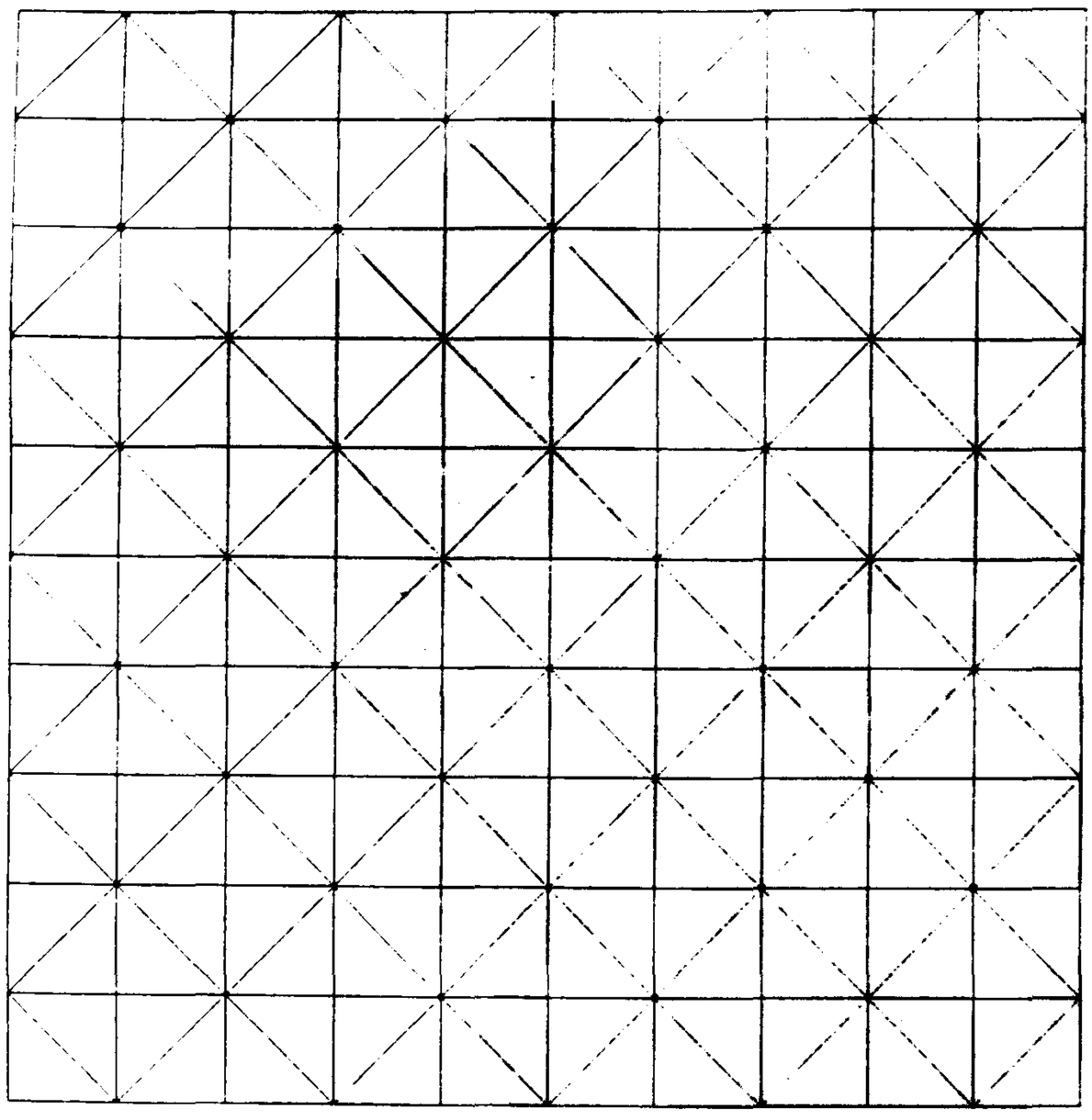


Figure 4.5 Coarse mesh.

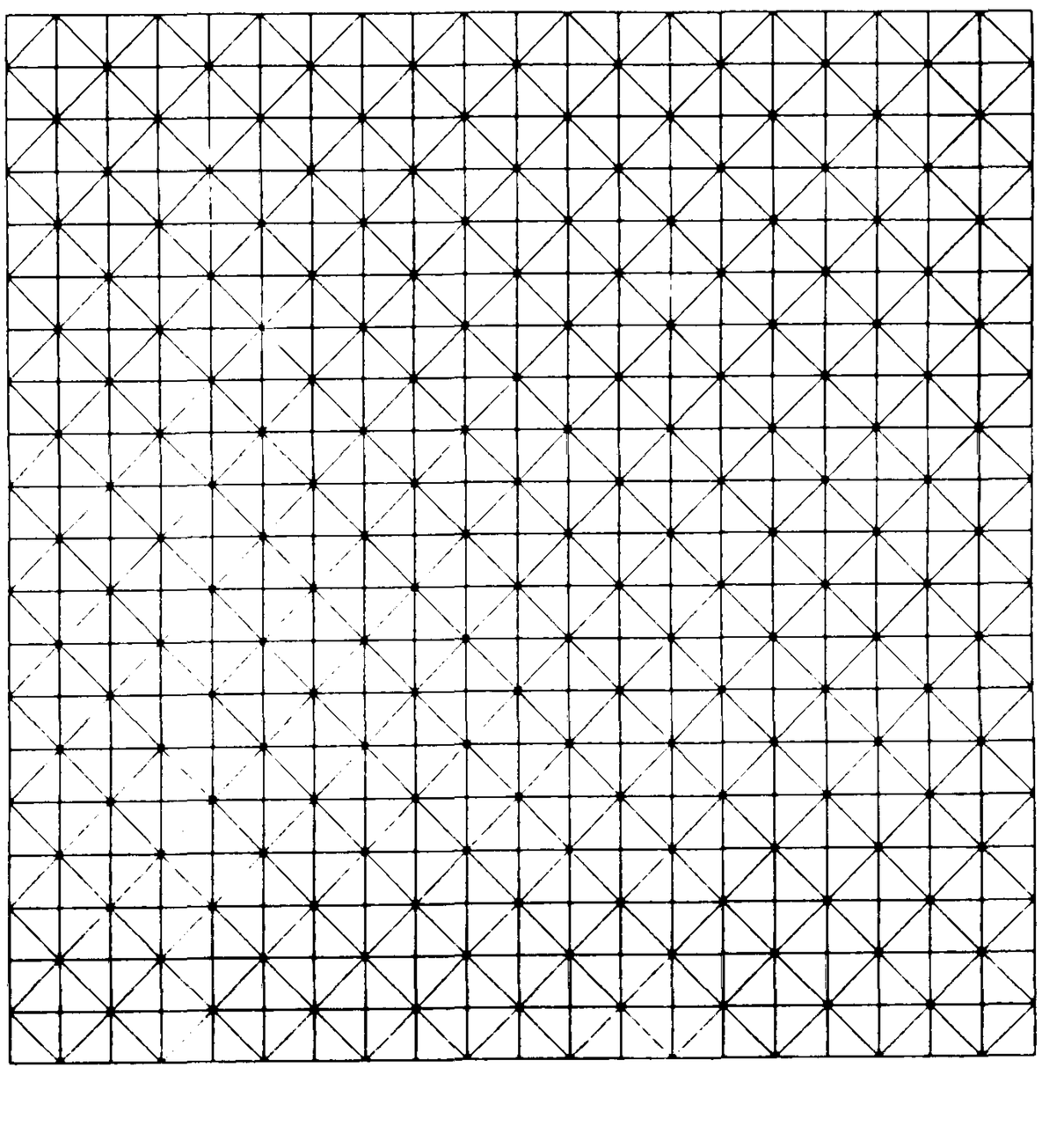
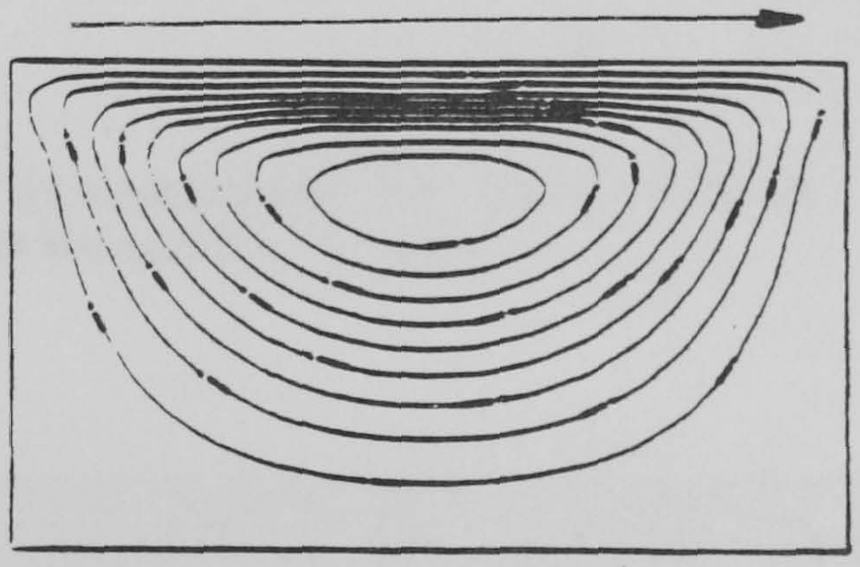
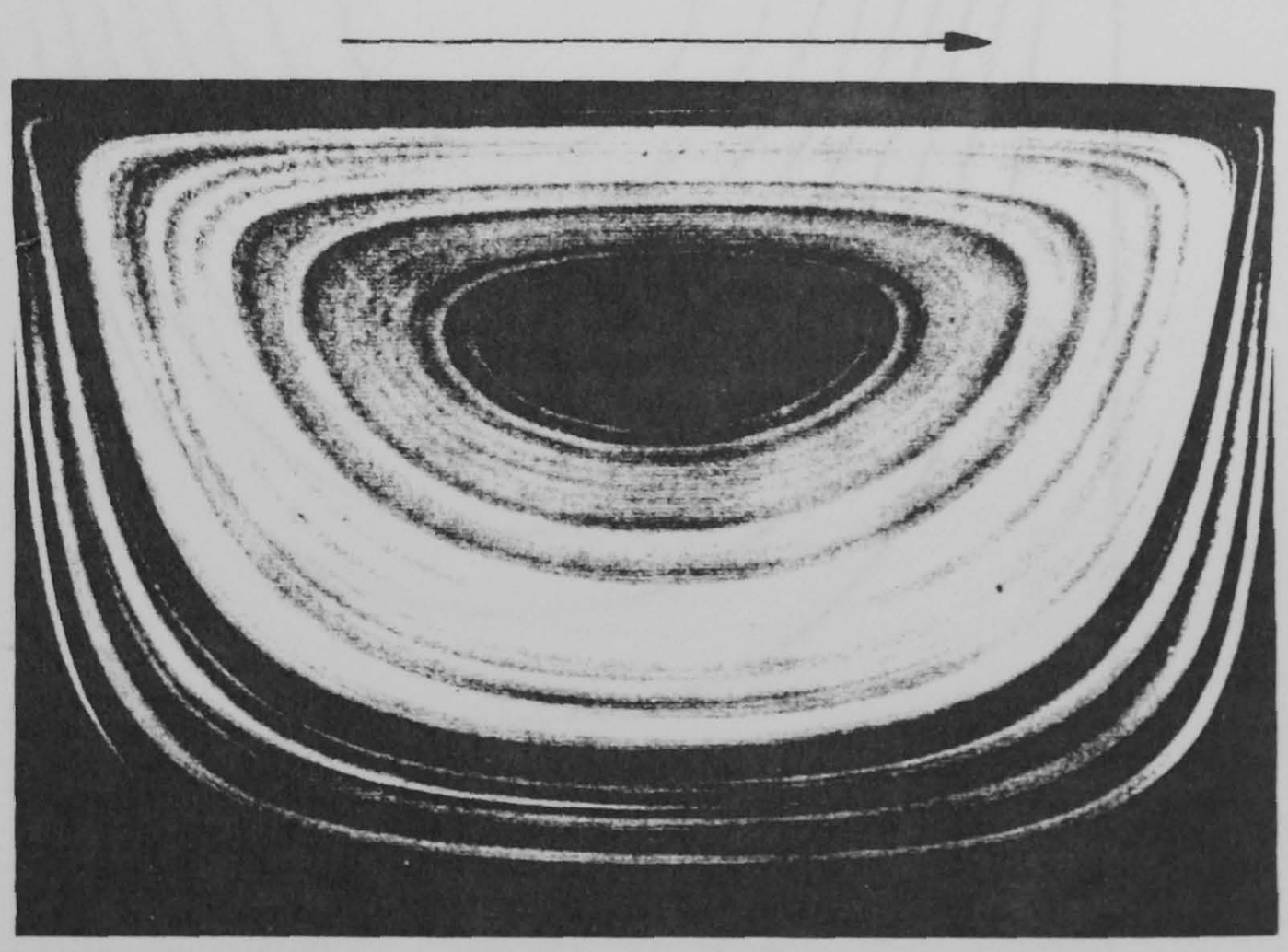


Figure 4.6 Refined mesh.

(a)



(b)



(c)

Figure 4.7 (a) Experimental streamlines at Re=0.6 by Chien et al.
(b) Theoretical streamlines at Re=0.6 by Chien et al.
(c) Experimental streamlines at Re=1.7 by Ottino.

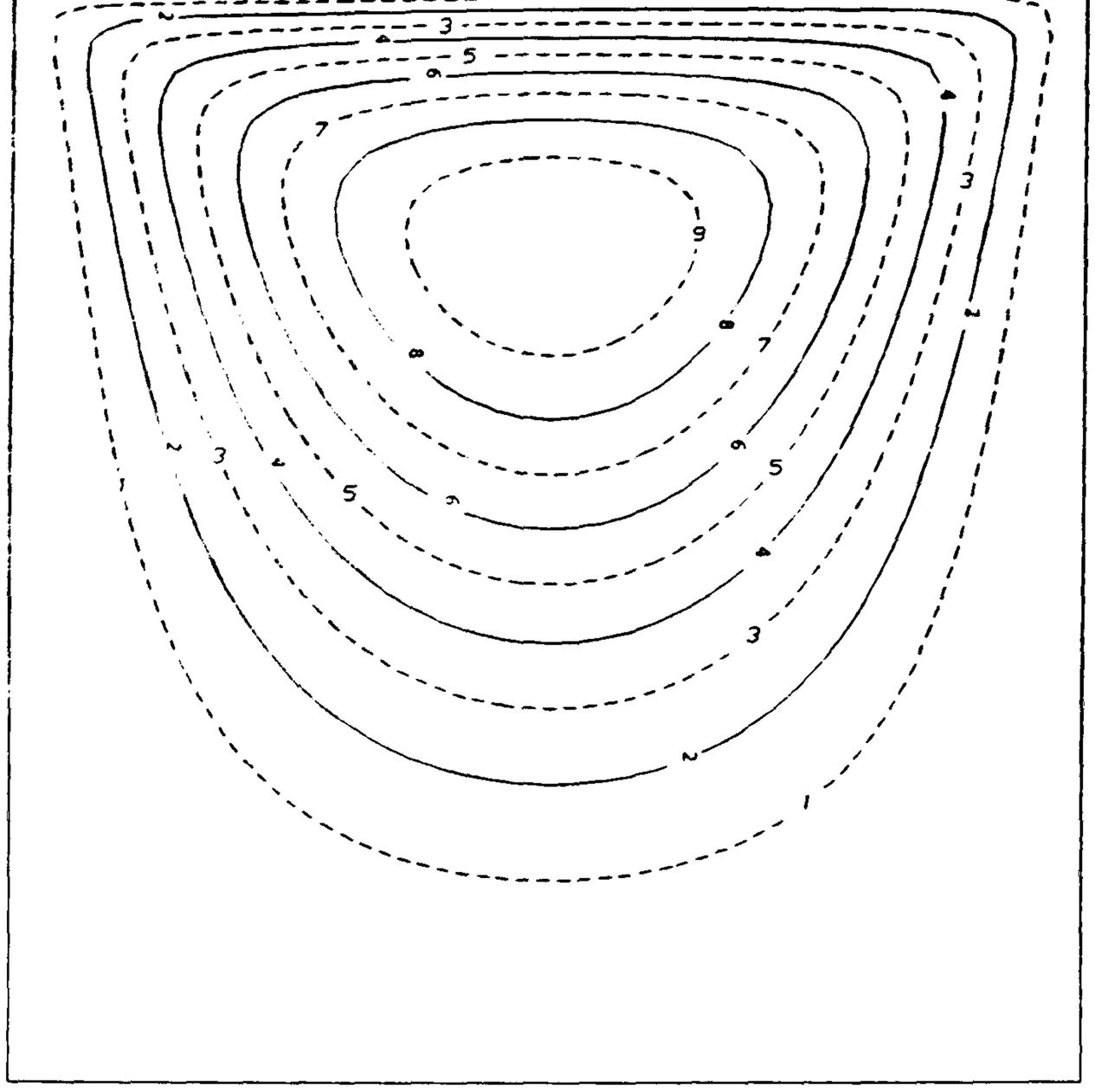


Figure 4.8 Streamlines at Re=1 for a Newtonian fluid.

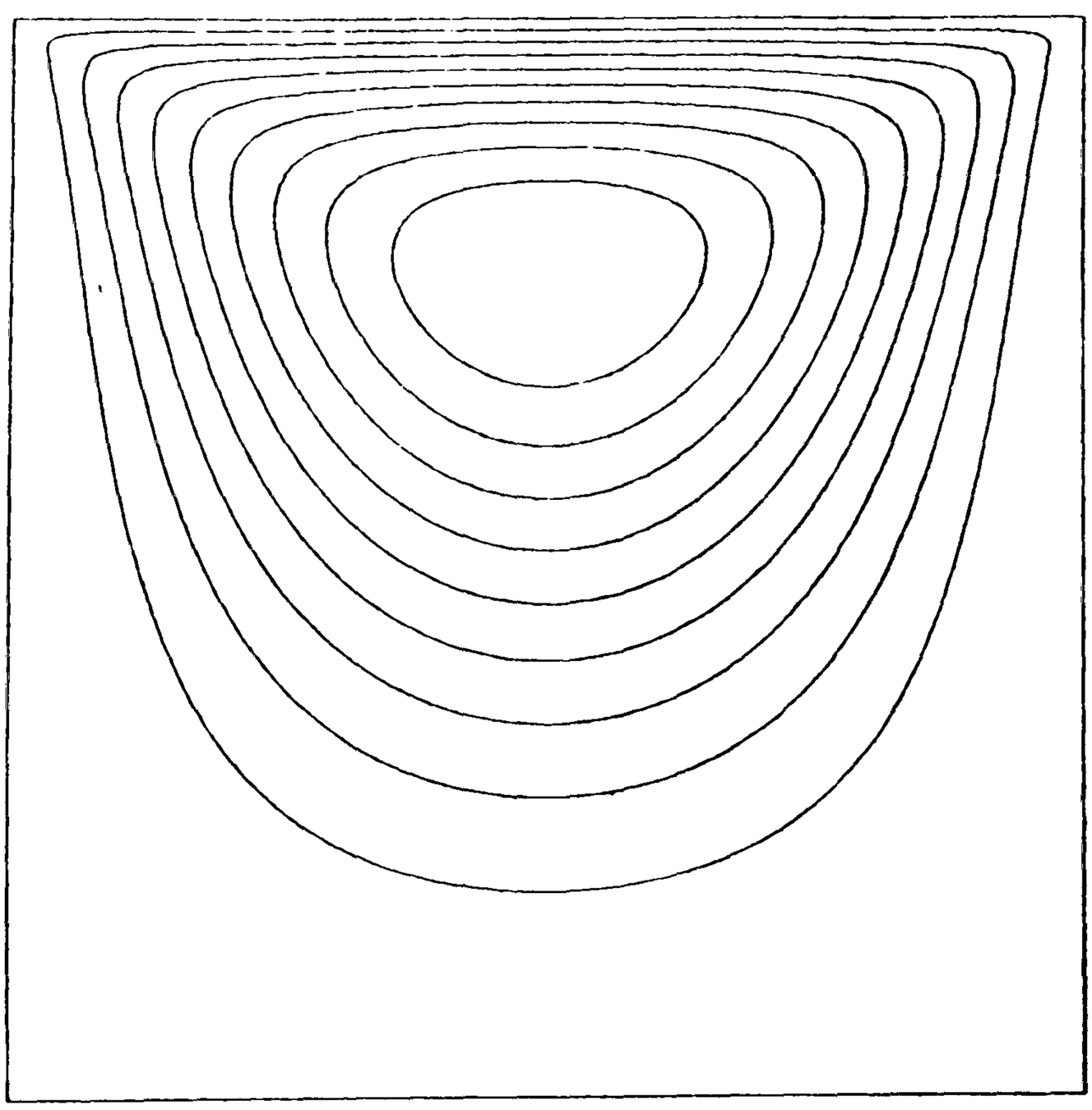


Figure 4.9 Streamfunction contours from -10^{-2} to -9×10^{-2} at intervals of 10^{-2} for $R_N = 1$ by Cliffe et al.

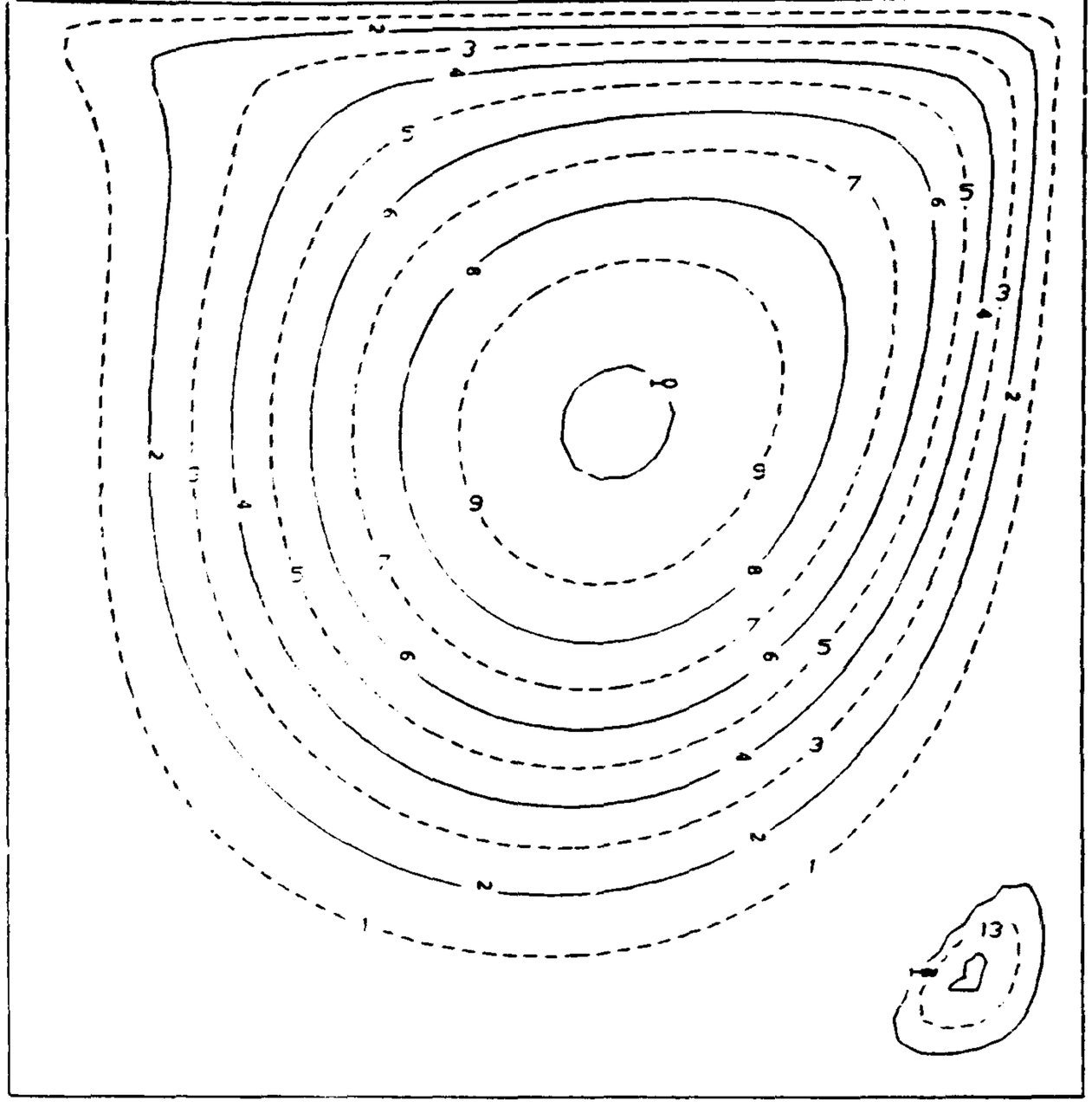


Figure 4.10 Streamlines at Re=400 for a Newtonian fluid.

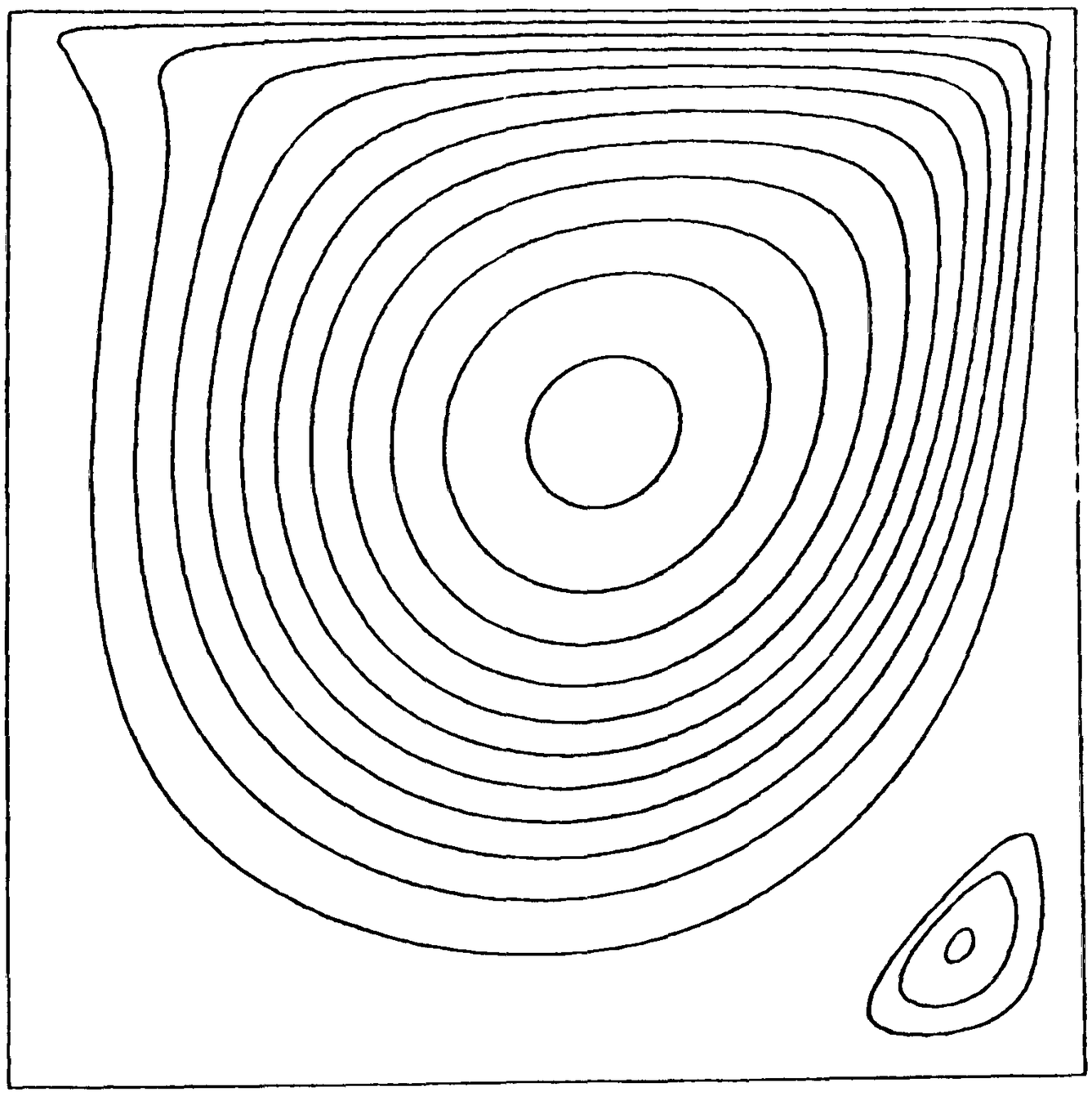
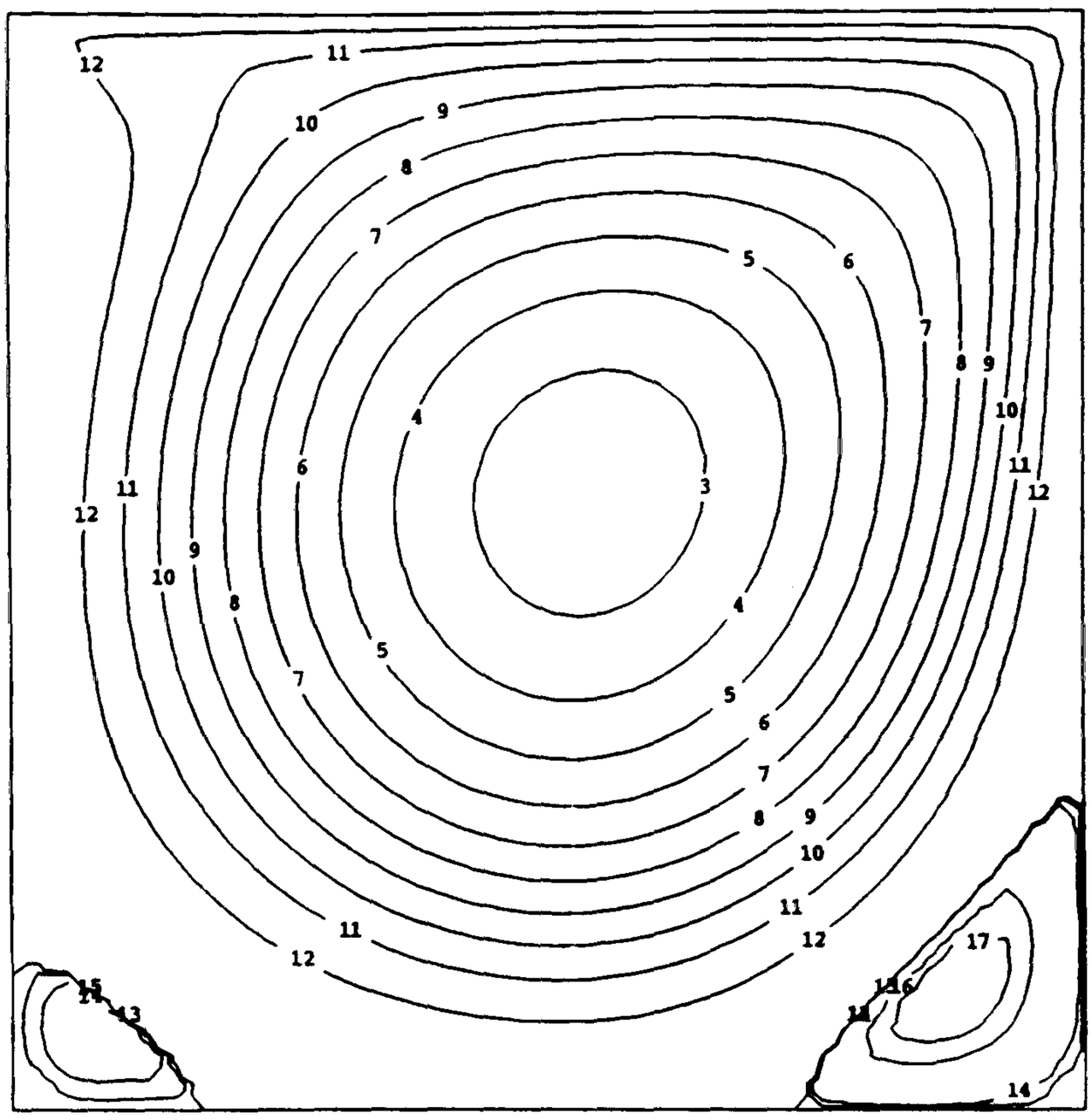


Figure 4.11 Streamfunction contours at R_N = 400: from -10^{-2} to -1.1×10^{-1} at intervals of 10^{-2} for primary eddy, and at 2, 4 and 6×10^{-4} for secondary eddy by Cliffe et al.



CONTOUR KEY	
1	-0.120E+00
2	-0.110E+00
3	-0.100E+00
4	-0.900E-01
5	-0.800E-01
6	-0.700E-01
7	-0.600E-01
8	-0.500E-01
9	-0.400E-01
10	-0.300E-01
11	-0.200E-01
12	-0.100E-01
13	0.000E+00
14	0.300E-04
15	0.100E-04
16	0.500E-03
17	0.800E-03

Figure 4.12 Streamlines at Re=1000 for a Newtonian fluid.

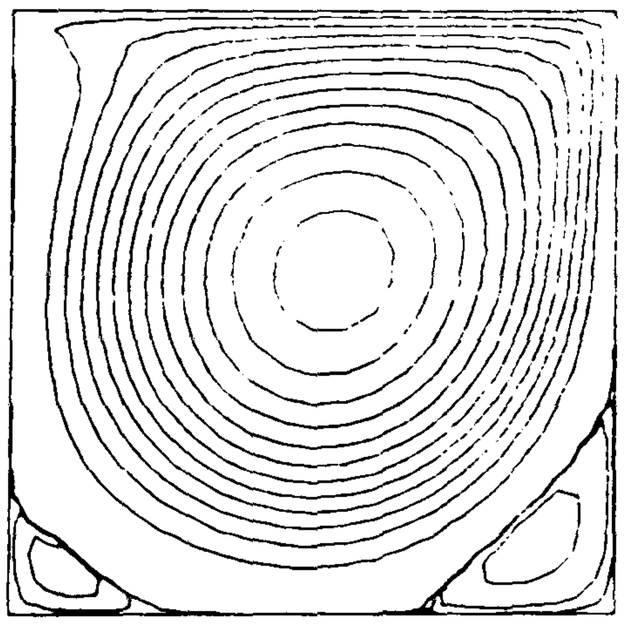
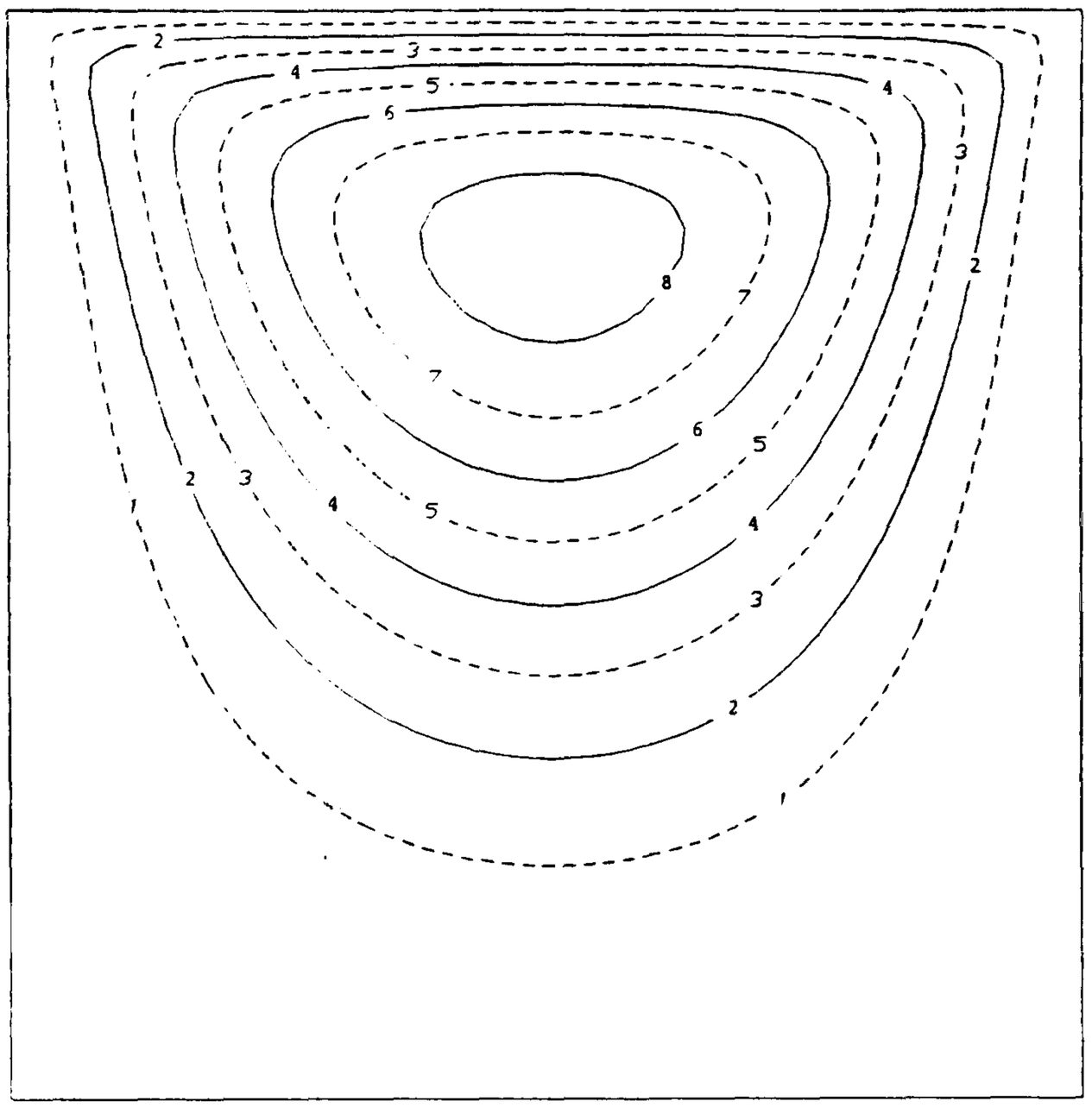
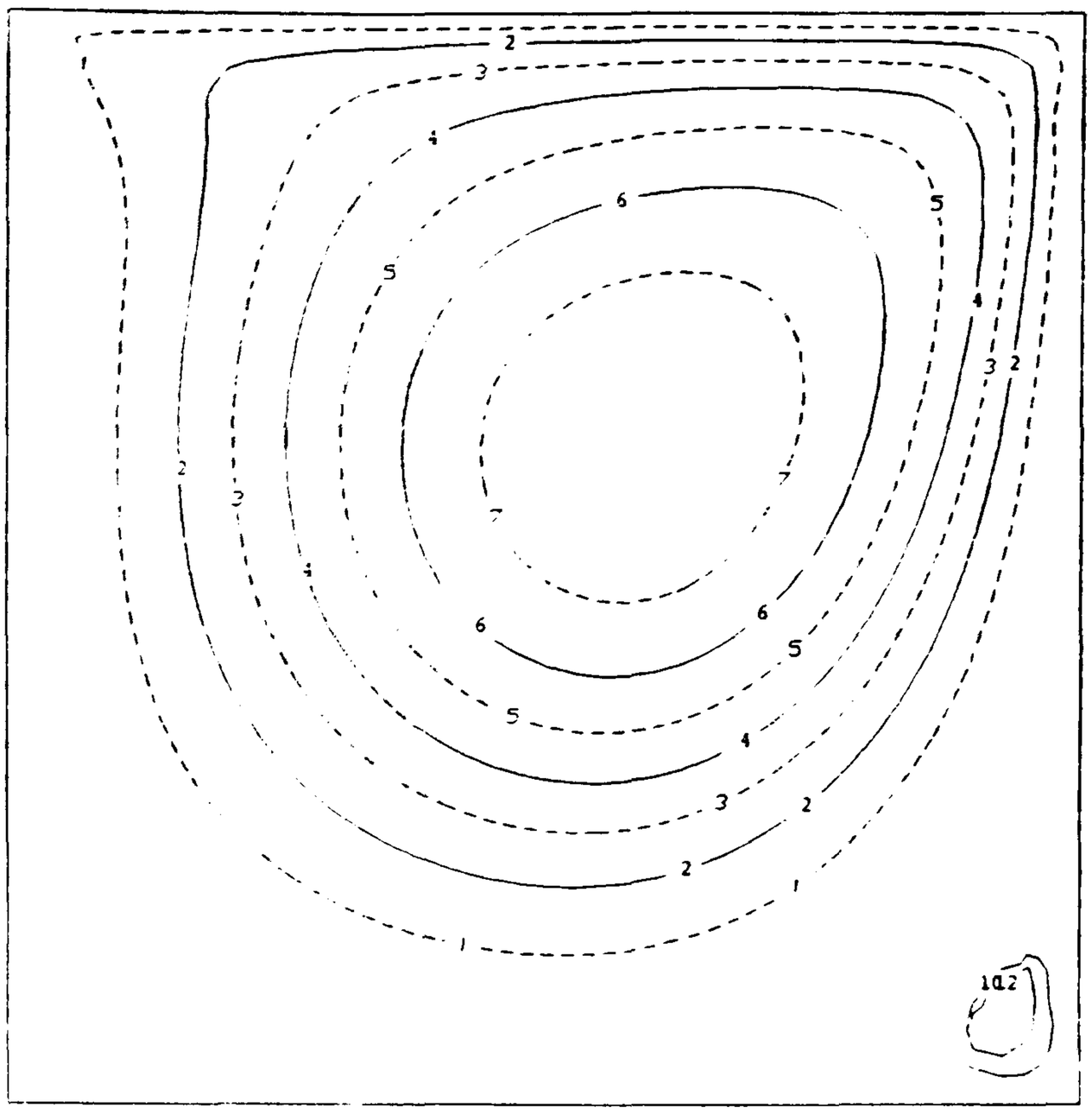


Figure 4.13 Streamlines at Re=1000 for a Newtonian fluid by Mizukami.



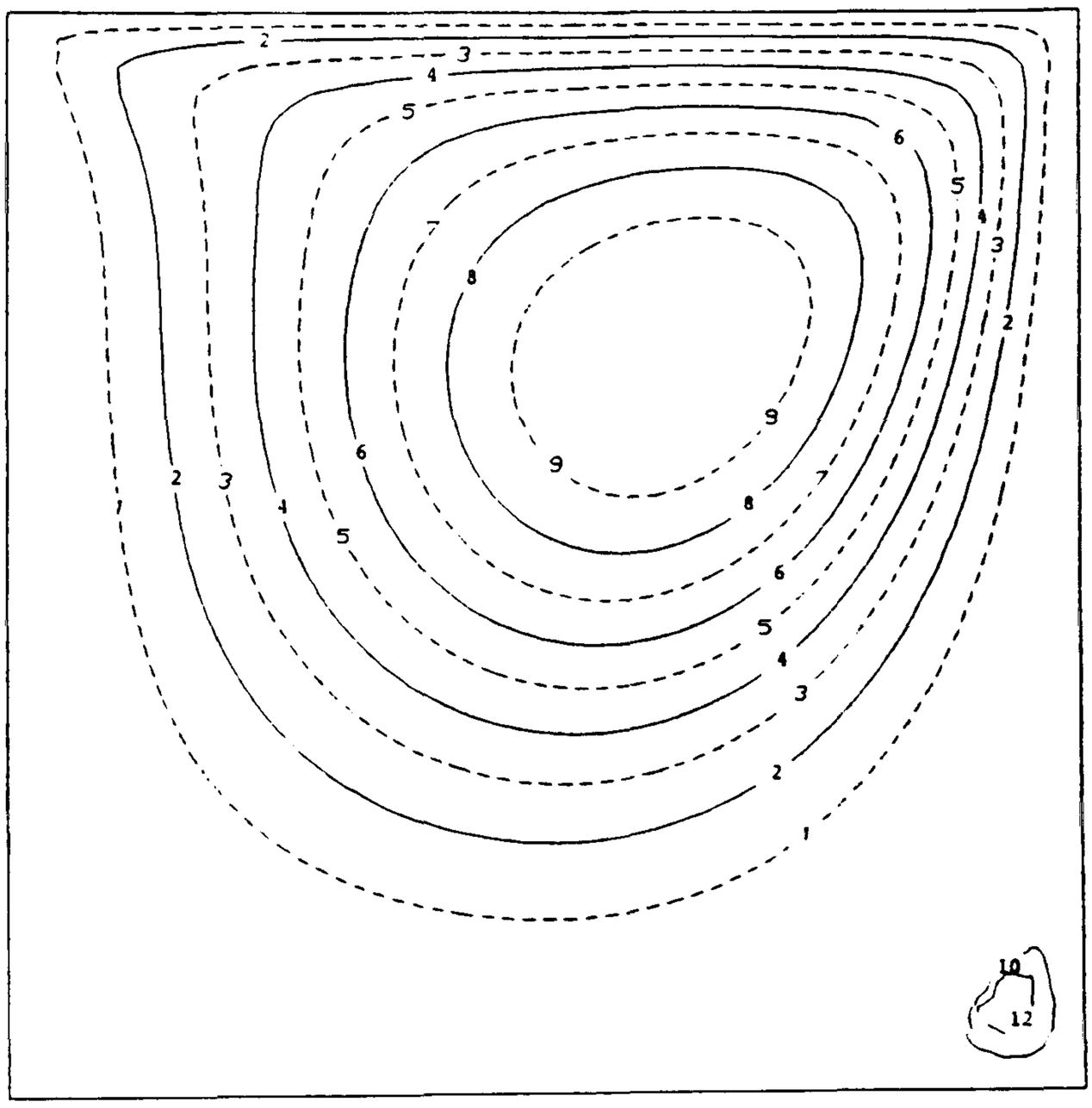
CONTOUR KEY	
1	-0.100E-01
2	-0.200E-01
3	-0.300E-01
4	-0.400E-01
5	-0.500E-01
6	-0.600E-01
7	-0.700E-01
8	-0.800E-01
9	-0.900E-01

Figure 4.14 Streamlines at Re=1 for a variable viscosity fluid.



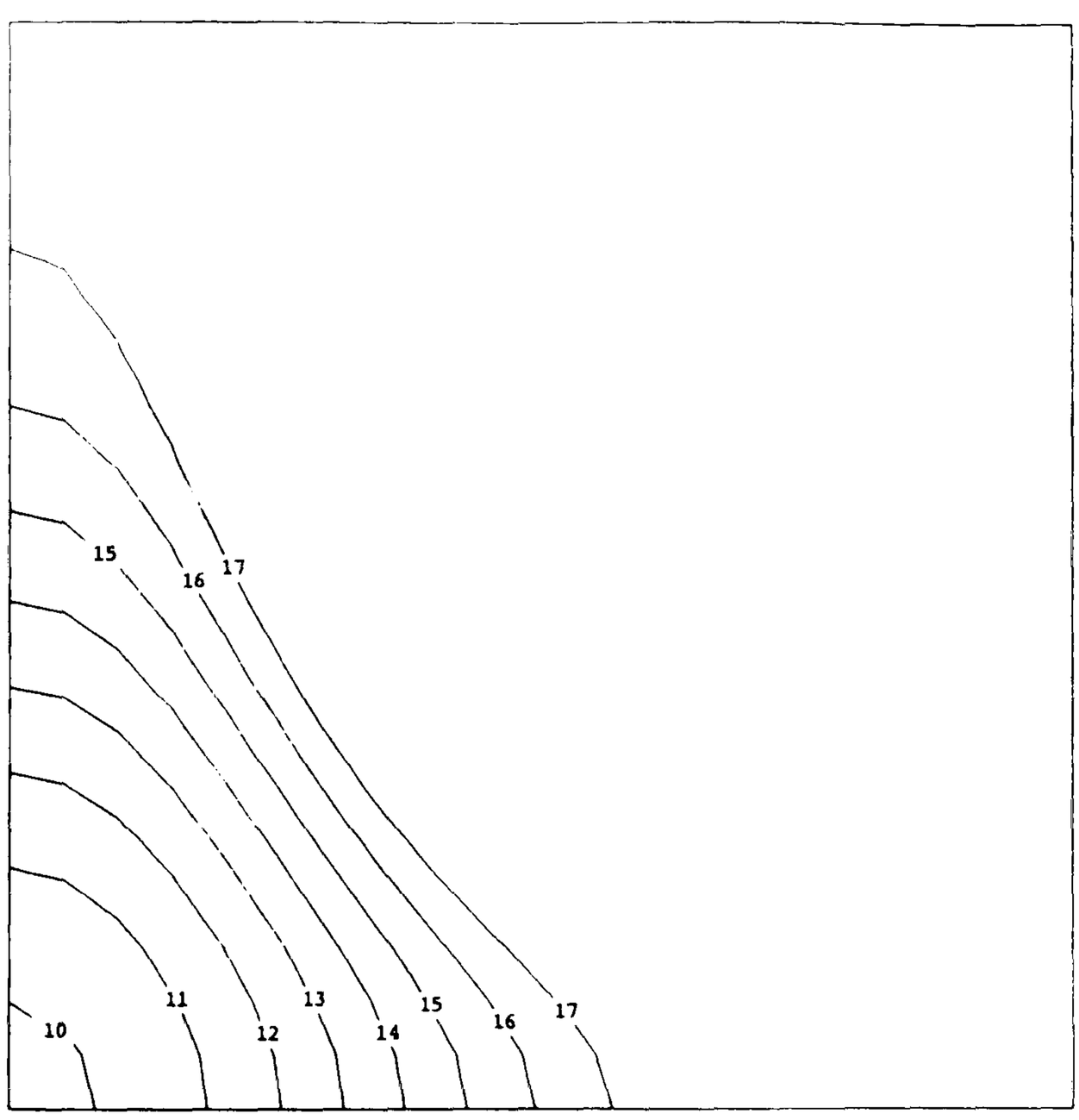
CONTOUR KEY	
1	-0.100E-01
2	-0.200E-01
3	-0.300E-01
4	-0.400E-01
5	-0.500E-01
6	-0.600E-01
7	-0.700E-01
8	-0.800E-01
9	-0.900E-01
10	0.000E+00
11	0.300E-04
12	0.100E-04

Figure 4.15 Streamlines at Re=200 for a variable viscosity fluid.



CONTOUR KEY	
1	-0.100E-01
2	-0.200E-01
3	-0.300E-01
4	-0.400E-01
5	-0.500E-01
6	-0.600E-01
7	-0.700E-01
8	-0.800E-01
9	-0.900E-01
10	0.000E+00
11	0.300E-04
12	0.100E-04

Figure 4.16 Streamlines at Re=200 for a Newtonian fluid.

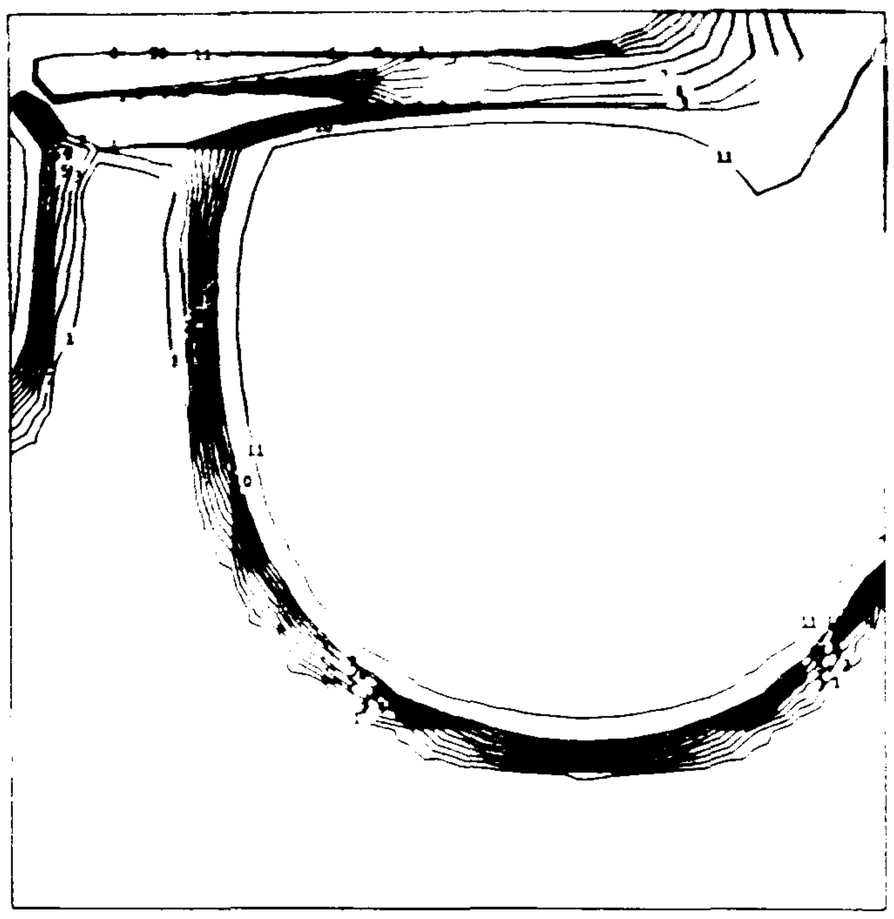


CONTOUR KEY	
1	0.100E+00
2	0.150E+00
3	0.200E+00
4	0.250E+00
5	0.300E+00
6	0.350E+00
7	0.400E+00
8	0.450E+00
9	0.500E+00
10	0.550E+00
11	0.600E+00
12	0.650E+00
13	0.700E+00
14	0.750E+00
15	0.800E+00
16	0.850E+00
17	0.900E+00

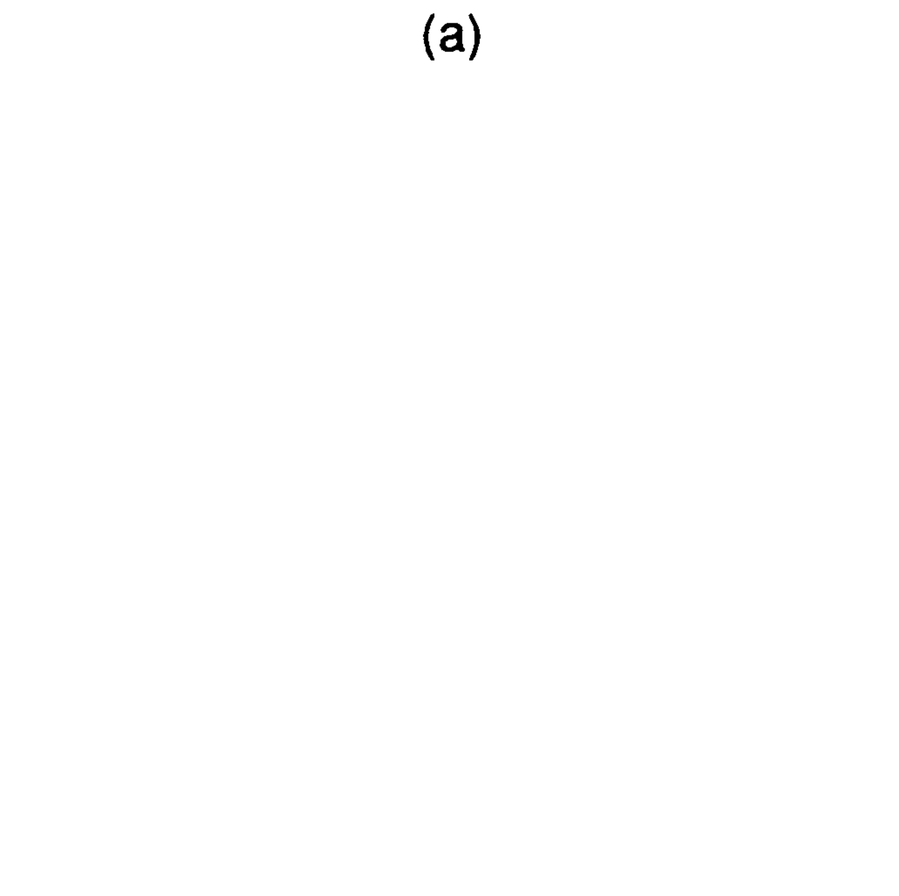
Figure 4.18 Concentration field at Re=1 for a Newtonian fluid after 5.2s (which is a 50% mix), produced by a non-upwinded finite element approach.

(a)

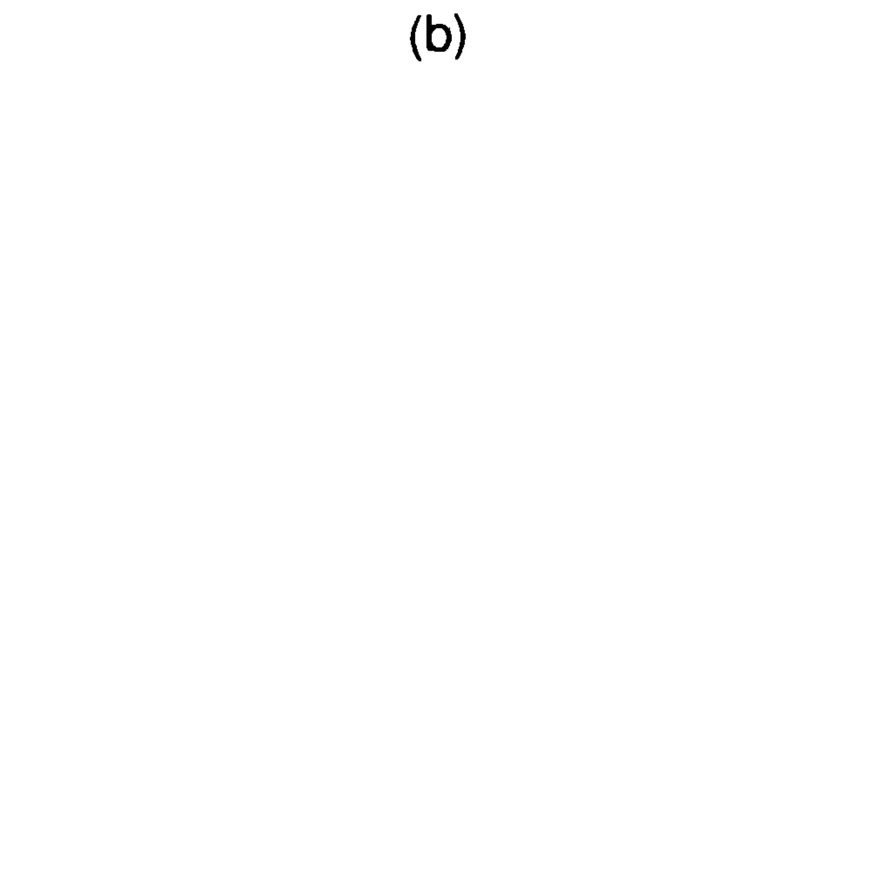
CONTOUR KEY	
1	0.100E+00
2	0.200E+00
3	0.300E+00
4	0.400E+00
5	0.500E+00
6	0.600E+00
7	0.700E+00
8	0.800E+00
9	0.900E+00
10	0.100E+01
11	0.150E+01



(b)



(c)



(d)

Figure 4.19 Concentration fields at Re=100 for a Newtonian fluid after; (a) 3 secs, (b) 4 secs, (c) 5 secs, (d) 6 secs. Results produced by an upwinded finite element approach.

CHAPTER 5

MATHEMATICAL SIMULATION OF AXISYMMETRIC MIXING PROBLEMS

5.1 Introduction

In this chapter, the numerical modelling for the prediction of flows due to stirrers rotating under both primary and secondary flow conditions is described. The Galerkin weak formulation U,V,W,P approach is used, this being an extended version of the U,V,P approach developed by Hughes and Taylor [42]. The flow problems considered are axisymmetric flows which although three dimensional in nature can be reduced to two dimensions. The way in which the finite element method is formulated for these problems is very similar to that described in Chapter 4.

5.2 Equations for steady flow

The flows that are of interest in this chapter are those produced by rotating a cylindrical stirrer, a rotating disc stirrer and also that of a rotating axisymmetric Rushton turbine in a cylindrical vessel, where the liquid is contained inside the vessel. The partial differential equations for this problem are specified in the following subsections :-

5.2.1 The Stress Equations Of Motion

The geometry suggests that the equations be defined in cylindrical polar co-ordinates, r,θ,z (fig. 5.1), with the z-axis along the axis of rotation. Only steady state isothermal flows are considered (i.e. $\partial/\partial t = 0$), with the density of the fluid taken as constant.

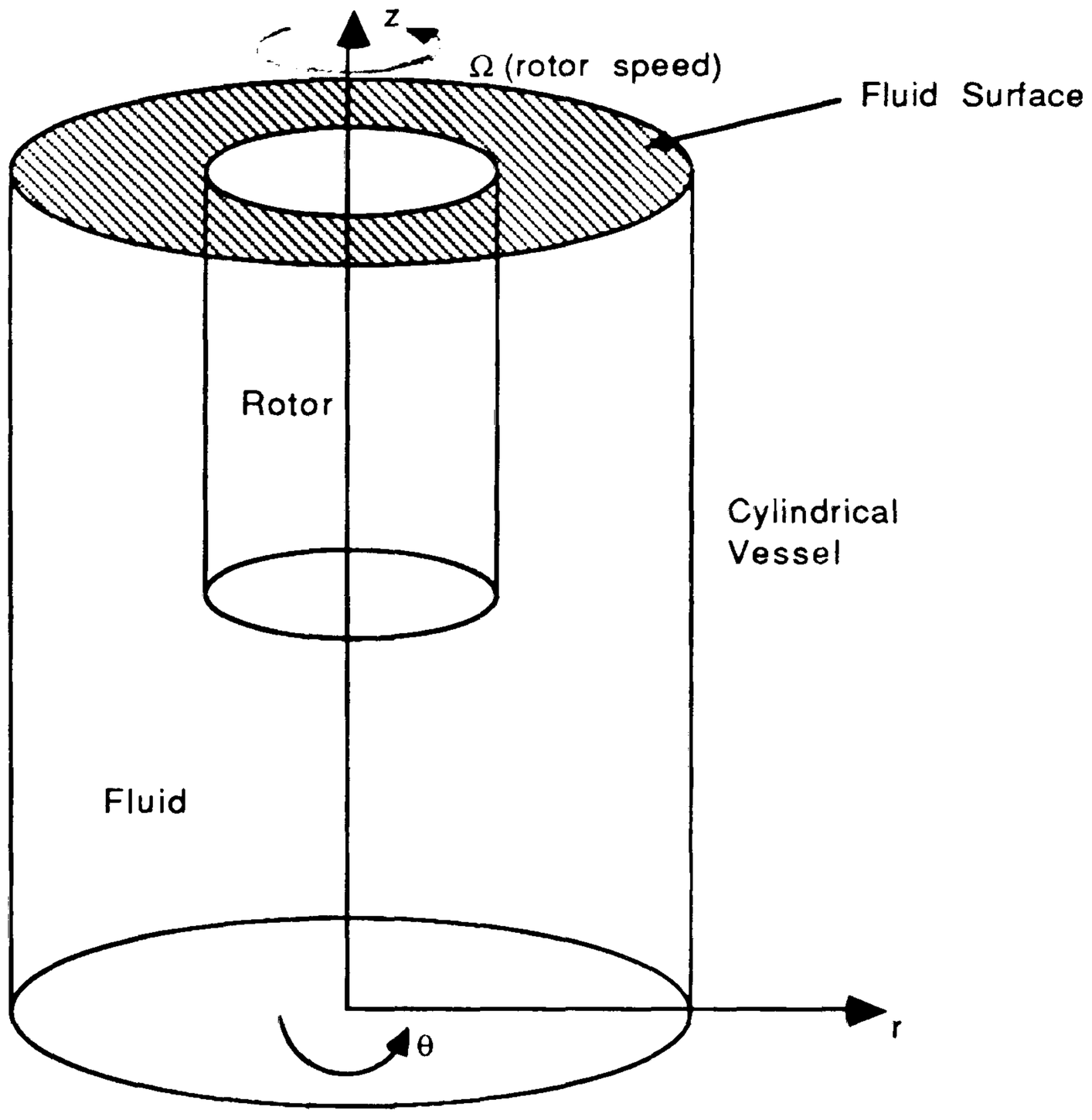


Figure 5.1 Diagram showing cylindrical co-ordinates.

As the flow is axisymmetric no changes occur in the θ direction and so $\partial/\partial\theta=0$. The three equations of momentum for this problem reduce to :-

$$-P_{,r} + (1/r)T_{rr} + T_{rr,r} + T_{rz,z} - (1/r)T_{\theta\theta} = \rho(UU_{,r} - V^2/r + WU_{,z}) \qquad (5.1)$$

$$(2/r)T_{r\theta} + T_{r\theta,r} + T_{\theta z,z} = \rho(UV_{,r} + UV/r + WV_{,z}) \qquad (5.2)$$

$$-P_{,z} + (1/r)T_{rz} + T_{rz,r} + T_{zz,z} - \rho g = \rho(UW_{,r} + WW_{,z}) \quad , \qquad (5.3)$$

where: T_{ij} are the physical components of the extra stress tensor due to the structure of the fluid, U,V,W are the velocities in the r,θ,z directions respectively, P is an arbitrary isotropic pressure, ρ is the density of the fluid and g is the gravitational constant.

5.2.2 The Continuity Equation

From the principle of conservation of mass for an incompressible fluid (and constant ρ) in three-dimensions the continuity equation is

$$U_{,r} + (1/r)U + W_{,z} = 0 \quad . \tag{5.4}$$

5.2.3 Rheological Equations Of State

These are also known as the constitutive equations and the particular equation

$$T_{ik} = 2\eta(q)E_{ik} \tag{5.5}$$

shows the relation between the stress T_{ik} and the deformation rate for a purely viscous generalised Newtonian fluid. E_{ik} is the first rate of strain tensor and represents velocity gradients with distance. From symmetry these rate of strain components reduce to six components hence the stress tensor is defined as follows :-

$$\begin{aligned}
T_{rr} &= 2\eta(q)E_{rr} = 2\eta(q)U_{,r} \\
T_{r\theta} &= 2\eta(q)E_{r\theta} = \eta(q)(V_{,r} - (1/r)V) \\
T_{rz} &= 2\eta(q)E_{rz} = \eta(q)(U_{,z} + W_{,r}) \\
T_{\theta\theta} &= 2\eta(q)E_{\theta\theta} = 2\eta(q)(1/r)U \\
T_{\theta z} &= 2\eta(q)E_{\theta z} = \eta(q)V_{,z} \\
T_{zz} &= 2\eta(q)E_{zz} = 2\eta(q)W_{,z} \quad ,
\end{aligned} \tag{5.6}$$

where q represents the deformation rate for the flow and is given by :-

$$q = \sqrt{[\ 2(E_{rr}^2 + E_{\theta\theta}^2 + E_{zz}^2 + 2(E_{r\theta}^2 + E_{rz}^2 + E_{\theta z}^2))\ ]} \ , \tag{5.7}$$

and therefore

$$q = \sqrt{[2(U_{,r})^2 + 2(U/r)^2 + 2(W_{,z})^2 + (V_{,r} - V/r)^2 + (U_{,z} + W_{,r})^2 + (V_{,z})^2\ ]}. \tag{5.8}$$

In (5.6) $\eta(q)$ represents the viscosity function, and in our problem the Cross model defined by equation (4.7) is used to model the variable-viscosity fluid.

5.3 Cylindrical Rotor

Since there is symmetry about the axis of rotation, only one half of the geometry need be considered for this problem (fig. 5.2). The base of the container (radius rb) is at z=0 and the base of the rotor (radius ra) is at z=za with the surface of the fluid at z=zb. The free surface at z=zb is assumed flat.

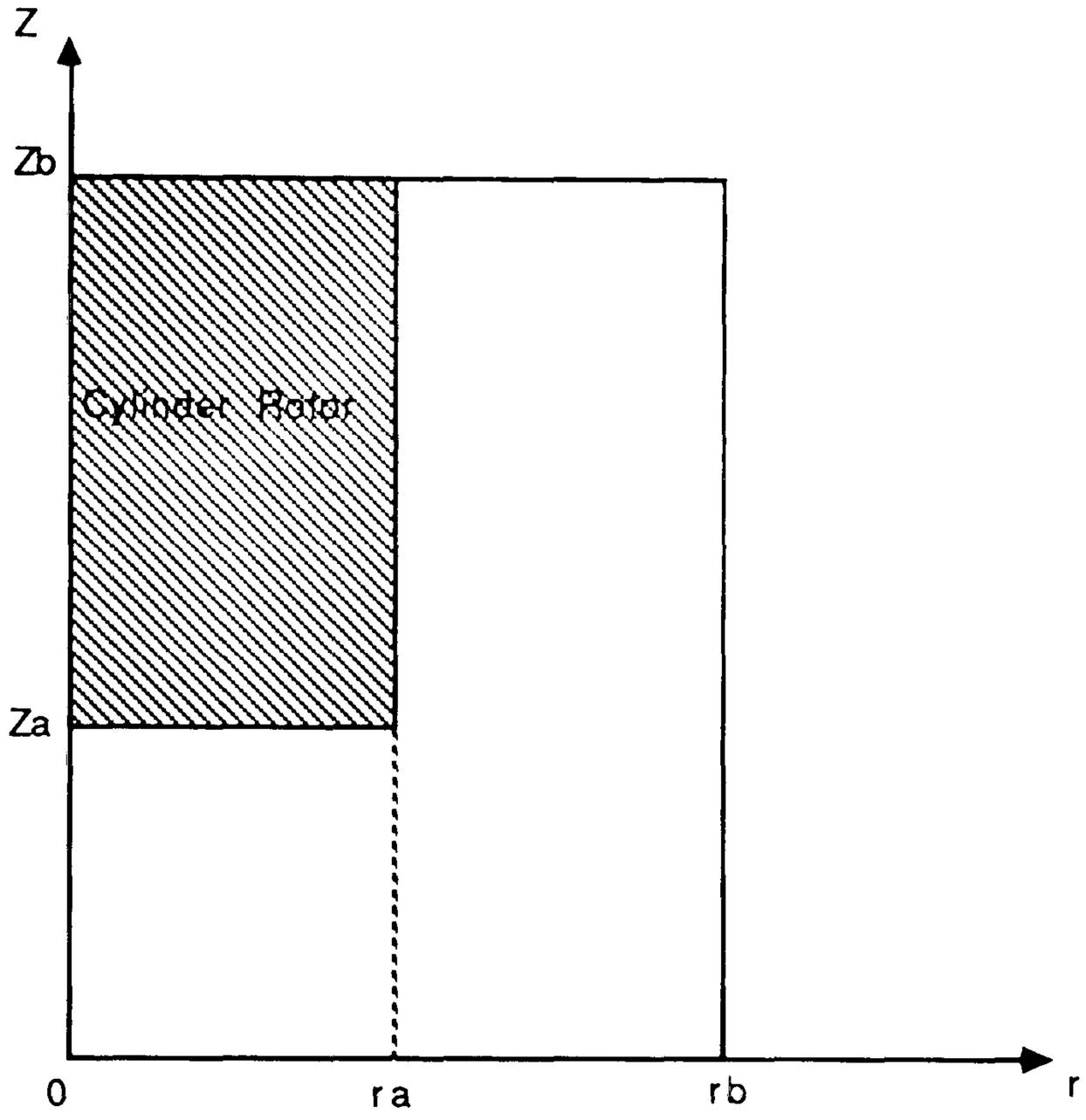


Figure 5.2 Diagram showing the axisymmetric representation of the cylindrical stirrer geometry.

5.3.1 Equations In Non-Dimensionalised Form

The equations are written in non-dimensional form to facilitate comparison between different fluids in different geometry dimensions from a common basis, and this is also useful for scaling up and down. In non-dimensionalised form the equations are,

$$-P_{,r} + (1/Re)T_{rr,r} + (1/Re)T_{rz,z} + (1/(r.Re))(T_{rr}-T_{\theta\theta}) - \rho(UU_{,r} - (1/r)V^2 + WU_{,z}) = 0 \qquad (5.9)$$

$$(1/Re)T_{r\theta,r} + (1/Re)T_{\theta z,z} + (2/(r.Re))T_{r\theta} - \rho(UV_{,r} + (1/r)UV + WV_{,z}) = 0 \qquad (5.10)$$

$$-P_{,z} + (1/Re)T_{rz,r} + (1/Re)T_{zz,z} + (1/(r.Re))T_{rz} - 1/Fr^2 - \rho(UW_{,r} + WW_{,z}) = 0 \qquad (5.11)$$

$$U_{,r} + (1/r)U + V_{,z} = 0 \quad , \qquad (5.12)$$

where $Re = \rho\, r_a^2 \Omega/\eta_o$ is the Reynolds number

$Fr = (Re/\eta_o \Omega g)^{1/2}$ is the Froude number

r_a is the rotor radius,

and Ω is the rotational speed.

These equations need modification for the special case when r=0, where there is an "apparent" coordinate singularity.

5.3.2 The U,V,W,P Formulation

The equations are formulated using the weak formulation of the Galerkin method in which, for axisymmetric flow, the integration must be performed over a three-dimensional domain surrounding the axis of symmetry, and we have

$$dA = 2\pi r\, d\bar{A} \qquad (5.13)$$

where $d\bar{s}$ is the differential element of area drxdz in a meridian plane. The boundary conditions are in general of the form:

Essential (Dirichlet);

$$U = Ua \;,\; V = Va \text{ and } W = Wa \quad \text{on } \Gamma_1 \qquad (5.14)$$

and

Natural (surface forces);

$$\tau_r = T_{rr}.n_r + T_{rz}.n_z$$
$$\tau_\theta = T_{r\theta}.n_r + T_{\theta z}.n_z \quad \text{on } \Gamma_2 \qquad (5.15)$$
$$\tau_z = T_{rz}.n_r + T_{zz}.n_z$$

where n_r, n_θ, n_z represent the outward normals in the r, θ and z directions respectively. Here $T_{rr}, T_{r\theta}, T_{rz}, T_{\theta\theta}, T_{\theta z}, T_{zz}$ are defined by (5.6) in non-dimensional form, also

$$\Gamma = \Gamma_1 + \Gamma_2$$

where Γ is the whole of the boundary of the domain.

Applying the Galerkin weighted residual method to the momentum and continuity equations, gives :

$$<-P_{,r} + (1/Re)(T_{rr,r} + T_{rz,z} + 1/r(T_{rr}-T_{\theta\theta})) - (UU_{,r} - V^2/r + WU_{,z}) ; 2\pi rS> = 0 \qquad (5.16)$$

$$< (1/Re)(T_{r\theta,r} + T_{\theta z,z} + (2/r)T_{r\theta}) - (UV_{,r} + UV/r + WV_{,z}) ; 2\pi rS> = 0 \qquad (5.17)$$

$$<-P_{,z} + (1/Re)(T_{rz,r} + T_{zz,z} + (1/r)T_{rz}) - 1/Fr^2 - (UW_{,r} + WW_{,z}) ; 2\pi rS> = 0 \qquad (5.18)$$

$$< U_{,r} + U/r + V_{,z} ; 2\pi rN> = 0 \qquad (5.19)$$

where S and N have the same meaning as defined in chapter 4 section 5. The weak formulation is formed by applying the divergence theorem to the stresses in the equations of momentum, and then substituting back into equations (5.16)-(5.18). To simplify the expression we will omit the factor of 2π and re-introduce it at the end of the formulation. We thus have

$$<P_{,r} ; rS> + <(1/Re)T_{rr} ; rS_{,r}> + <(1/Re)T_{rz} ; rS_{,z}> + <(1/Re)T_{\theta\theta} ; S> + <UU_{,r} + WU_{,z} ; rS> - <V^2 ; S> = \int_{\Gamma_2} \tau_r.rS \, ds \qquad (5.20)$$

$$<(1/Re)T_{r\theta} ; rS_{,r}> + <(1/Re)T_{\theta z} ; rS_{,z}> - <(1/Re)T_{r\theta} ; S> + <UV_{,r} + WV_{,z} ; rS> + <UV ; S> = \int_{\Gamma_2} \tau_\theta.rS \, ds \, . \qquad (5.21)$$

$$<P_{,z} ; rS> + <(1/Re)T_{rz} ; rS_{,r}> + <(1/Re)T_{zz} ; rS_{,z}> + <UW_{,r} + WW_{,z} ; rS> = \int_{\Gamma_2} \tau_z.rS \, ds - <1/Fr^2 ; rS> \qquad (5.22)$$

$$< U_{,r} + U/r + V_{,z} ; rN > = 0 \qquad (5.23)$$

Consider a typical element with U,V,W and P approximated by

$$\tilde{U}=\sum_{i=1}^{m} U_i.S_i \;;\quad \tilde{V}=\sum_{i=1}^{m} V_i.S_i \;;\quad \tilde{W}=\sum_{i=1}^{m} W_i.S_i \;;\quad \tilde{P}=\sum_{l=1}^{b} P_l.N_l \qquad (5.24)$$

where S_i and N_l are shape functions as defined in section 4.5, then S and N must satisfy (5.20) - (5.23) for all values of i and l. The four equations above can now be written as

$$\langle \tilde{P}_{,r} ; rS_i\rangle + \langle (2\eta(\tilde{q})/Re)\tilde{U}_{,r} ; rS_{i,r}\rangle + \langle (\eta(\tilde{q})/Re)(\tilde{U}_{,z} + \tilde{W}_{,r}) ; rS_{i,z}\rangle$$
$$+ \langle (2\eta(\tilde{q})/Re)\tilde{U}/r ; S_i\rangle + \langle \tilde{U}\tilde{U}_{,r} + \tilde{W}\tilde{U}_{,z} ; rS_i\rangle - \langle \tilde{V} ; S_i\rangle = \int_{\Gamma_2} \tilde{\tau}_r.rS_i \, ds \qquad (5.25)$$

$$\langle (\eta(\tilde{q})/Re)(\tilde{V}_{,r} - \tilde{V}/r) ; rS_{i,r}\rangle + \langle (\eta(\tilde{q})/Re)\tilde{V}_{,z} ; rS_{i,z}\rangle$$
$$- \langle (\eta(\tilde{q})/Re)(\tilde{V}_{,r} - \tilde{V}/r) ; S_i\rangle + \langle \tilde{U}\tilde{V}_{,r} + \tilde{W}\tilde{V}_{,z} ; rS_i\rangle + \langle \tilde{U}\tilde{V} ; S_i\rangle = \int_{\Gamma_2} \tilde{\tau}_\theta.rS_i \, ds \qquad (5.26)$$

$$\langle \tilde{P}_{,z} ; rS_i\rangle + \langle (\eta(\tilde{q})/Re)(\tilde{U}_{,z} + \tilde{W}_{,r}) ; rS_{i,r}\rangle + \langle (2\eta(\tilde{q})/Re)\tilde{W}_{,z} ; rS_{i,z}\rangle$$
$$+ \langle \tilde{U}\tilde{W}_{,r} + \tilde{W}\tilde{W}_{,z} ; rS_i\rangle = \int_{\Gamma_2} \tilde{\tau}_z.rS_i \, ds - \langle 1/Fr^2 ; rS_i\rangle \qquad (5.27)$$

$$\langle \tilde{U}_{,r} + \tilde{U}/r ; rN_l\rangle + \langle \tilde{V}_{,z} ; rN_l\rangle = 0 \qquad (5.28)$$

where $1 <= i <= m$ and $1 <= l <= b$. In matrix component form these are as follows on re-introduction of the 2π term,

$$\begin{aligned}
&\mathbf{A1}_{ij}U_j + \mathbf{A3}_{ij}W_j + \mathbf{A4}_{il}P_l + (\mathbf{A9}_{ijk}U_j + \mathbf{A10}_{ijk}W_j)U_k - \mathbf{A11}_{ijk}V_jV_k = X_i \\
&\mathbf{A8}_{ij}V_j + (\mathbf{A9}_{ijk}U_j + \mathbf{A10}_{ijk}W_j)V_k + \mathbf{A11}_{ijk}V_kU_j = Y_i \qquad (5.29) \\
&\mathbf{A3}_{ji}U_j + \mathbf{A2}_{ij}W_j + \mathbf{A5}_{in}P_n + (\mathbf{A9}_{ijk}U_j + \mathbf{A10}_{ijk}W_j)W_k = Z_i \\
&\mathbf{A6}_{lj}U_j + \mathbf{A7}_{lj}W_j = 0
\end{aligned}$$

where:

$$\mathbf{A1}_{ij} = \langle 2(\eta(\tilde{q})/Re)S_{j,r} ; 2\pi rS_{i,r}\rangle + \langle (\eta(\tilde{q})/Re)S_{j,z} ; 2\pi rS_{i,z}\rangle$$
$$+ \langle 2(\eta(\tilde{q})/Re)S_{j/r} ; 2\pi S_i\rangle \,,$$

$$\mathbf{A2}_{ij} = \langle (\eta(\tilde{q})/Re)S_{j,r} ; 2\pi rS_{i,r}\rangle + \langle 2(\eta(\tilde{q})/Re)S_{j,z} ; 2\pi rS_{i,z}\rangle \,,$$

$$\mathbf{A3}_{ij} = \langle (\eta(\tilde{q})/Re)S_{j,r} ; 2\pi rS_{i,z}\rangle \,,$$

$$\mathbf{A4}_{il} = \langle N_{l,r} \; ; \; 2\pi r S_i \rangle \quad ,$$

$$\mathbf{A5}_{il} = \langle N_{l,z} \; ; \; 2\pi r S_i \rangle \quad ,$$

$$\mathbf{A6}_{lj} = \langle r S_{j,r} + S_j \; ; \; 2\pi N_l \rangle \quad ,$$

$$\mathbf{A7}_{lj} = \langle S_{j,z} \; ; \; 2\pi r N_l \rangle \quad ,$$

$$\mathbf{A8}_{ij} = \langle (\eta(\tilde{q})_{/Re}) S_{j,r} \; ; \; 2\pi r S_{i,r} \rangle - \langle (\eta(\tilde{q})_{/Re}) S_j \; ; \; 2\pi S_{i,r} \rangle$$

$$+ \langle (\eta(\tilde{q})_{/Re}) S_{j,z} \; ; \; 2\pi r S_{i,z} \rangle - \langle (\eta(\tilde{q})_{/Re}) S_{j,r} \; ; \; 2\pi S_i \rangle$$

$$+ \langle (\eta(\tilde{q})_{/Re}) S_{j/r} \; ; \; 2\pi S_i \rangle \quad ,$$

$$\mathbf{A9}_{ijk} = \langle S_j S_{k,r} \; ; \; 2\pi r S_i \rangle \quad ,$$

$$\mathbf{A10}_{ijk} = \langle S_j S_{k,z} \; ; \; 2\pi r S_i \rangle \quad ,$$

$$\mathbf{A11}_{ijk} = \langle S_j S_k \; ; \; 2\pi S_i \rangle \quad ,$$

with the vector components being :

$$X_i = \int_{\Gamma_2} \tilde{\tau}_r . 2\pi r . S_i \, ds \quad ,$$

$$Y_i = \int_{\Gamma_2} \tilde{\tau}_\theta . 2\pi r . S_i \, ds \quad ,$$

$$Z_i = \int_{\Gamma_2} \tilde{\tau}_z . 2\pi r . S_i \, ds \quad - \quad (1/Fr^2) \langle 1 \; ; \; 2\pi r S_i \rangle \quad ,$$

and $1 <= i,j,k <= m$, $1 <= l <= b$. This algebraic system has to be calculated for every element in the mesh and added into its respective positions in the overall system matrix. This system is non-linear in the convective terms and in the nature of the fluid when considering the non-Newtonian case. These non-linearities are accommodated by linearising the problem in a similar way to that explained in section 4.5.

The system matrix is again non-positive definite so the frontal elimination solver is used. Due to the reasons given in section 4.5 triangular elements similar to fig 4.4 are used, with the addition that velocity component W along with U, V are specified at the six nodal points of the triangle. A far simpler set of equations were obtained for the case r=0.

5.4 Programs Developed To Solve Axisymmetric Flow Problems

The programs that had been written for the 2-D problem of the previous chapter, were modified and parts completely re-written for the axisymmetric case. Therefore only the major modifications are mentioned.

5.4.1 Finite Element Program

Changes were made to the element matrix assembly routine, these alterations involved the implementation of the algebraic system (5.29).

5.4.2 Stream Function Program

The equation for the stream function is different for the axisymmetric case, with the relationship between the stream function ϕ and the velocities as follows :

$$rU = \phi_{,z} \tag{5.30}$$

$$rW = -\phi_{,r} \tag{5.31}$$

Differentiating the first equation with z and the second with r and combining both equations together we obtain the following Poisson equation :

$$\phi_{,zz} + \phi_{,rr} = r(U_{,z} - W_{,r}) - W \tag{5.32}$$

This equation is solved in the same way as for the two dimensional case with an addition to the contour routine of an option for drawing contour plots of rV values as well as the stream function for comparing with finite difference work.

5.5 Results For The Cylindrical Rotor

For this problem the dimensions of the geometry and the applied boundary conditions are shown in figure 5.3. Two rotor geometry problems are considered in this section for comparative studies.

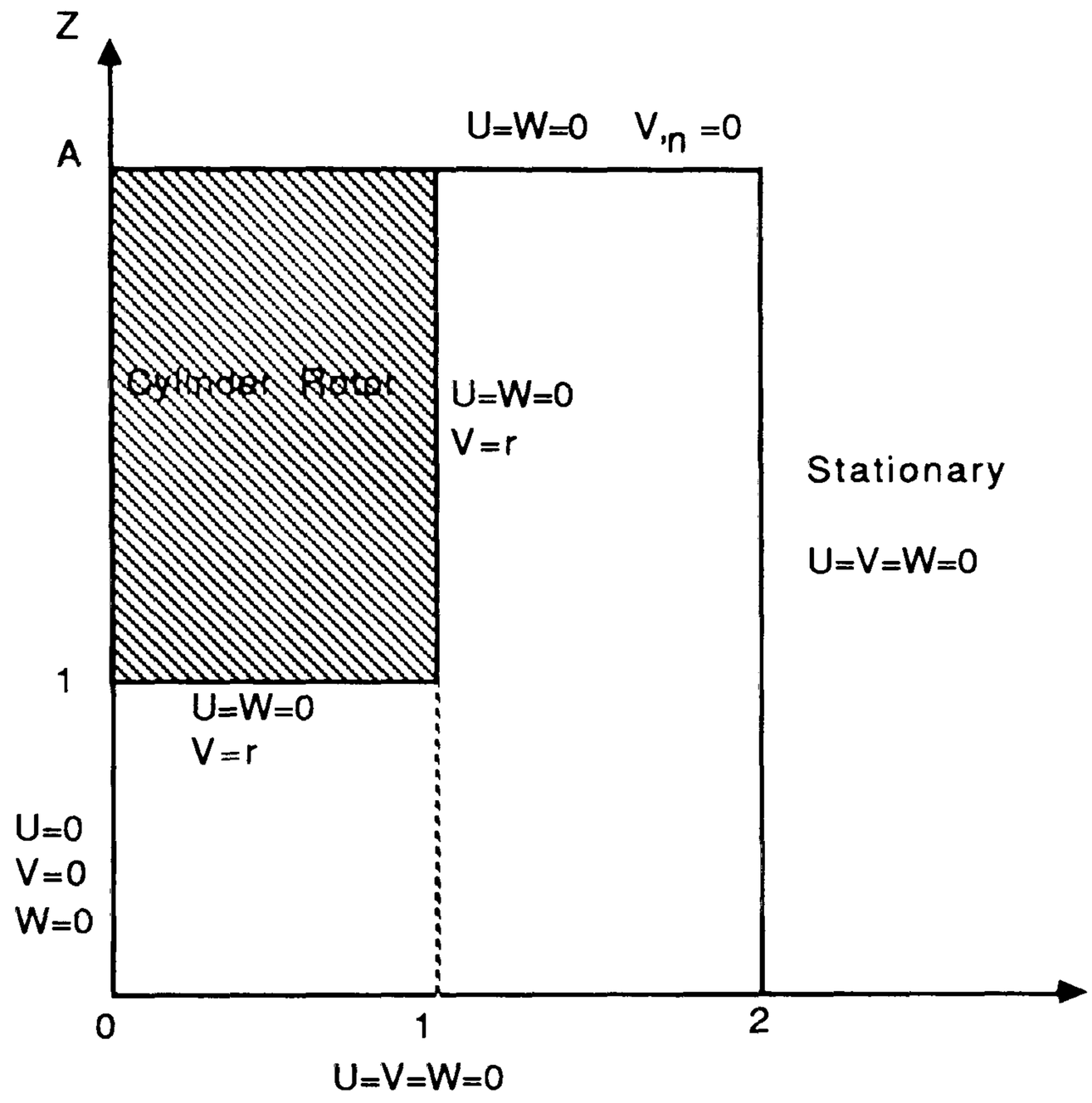


Figure 5.3 Diagram showing dimensions and boundary conditions for the cylindrical stirrer.

5.5.1 The Short Geometry

The geometry is shown in fig. 5.3 with A taking a value of two. We used three meshes for the problem. Firstly we defined a fairly coarse mesh (fig. 5.4*) of 225 nodes. Due to the rough nature of the streamlines in computed results and the inability to reach high Reynolds numbers we saw the need for a greater overall refinement of the mesh and produced a refined second version (fig. 5.5*) which had 833 nodes. Thirdly the mesh was refined only in regions of particular interest (fig. 5.6*) using 1281 nodes. To justify the mathematical method used for this study, we have compared our results with those of other workers.

5.5.1.1 The Work Of Bodalia [6]

Bodalia used the same geometry as shown in Fig. 5.3. Her numerical results were obtained by using the finite difference method to solve the cylindrical rotor problem, for both Newtonian and Cross type fluids. In figures 5.7*-5.10* comparison is made with streamline and rV plots at Re=100 for a Newtonian fluid, and it is apparent that the general pattern is very similar. Streamline and rV plots at Re=100 for the Cross fluid are also compared (fig 5.11* - 5.14*). It is clearly seen that these are again very similar.

From this we can conclude that results produced by the finite element and finite difference programs are in good agreement even though a coarser mesh was employed for the finite difference calculations.

5.5.1.2 The Work Of Lugt And Abboud [67]

Lugt and Abboud considered the flow circulation in a closed cylindrical container which was produced by a rotating lid, the fluid used was Newtonian in nature, and the numerical method applied to the problem was the finite difference method. Axisymmetric vortex breakdown on centreline occured at values of Reynolds number greater than Re =1000, (fig 5.15*) was shown,and the appearance of a stagnation point on the axis of symmetry was explained by the pressure increase due to diverging helical streamlines and not due to instability.

Presented in their report was a photograph by Escudier [30] of this phenomenon (fig 5.16*) at Re=1850 with d=2, where d is the ratio of the height of the vessel to its radius. In their paper results were given for values of d between 1.5 and 2. Vogel [110] has obtained results for d=1.58 which showed the appearance of vortex breakdown at a much lower Reynolds number, Re=1130.

For the geometry d=1 presented here (i.e. the height of the vessel is 2 and the radius is 2) axisymmetric vortex breakdown has already appeared by Re=1000 (fig 5.17*) with one stagnation point occuring on the axis of symmetry. From the work by others discussed above it may be inferred that as the d number becomes smaller vortex breakdown occurs earlier, at a lower Reynolds number, even less than Re=1000, which agrees with our results.

5.5.2 The Tall Geometry

The geometry is shown in fig. 5.3 with A taking the value 5. In this problem the height of the cylindrical rotor is large in comparison to the gap width between the rotor and the fixed outer wall and in most of this narrow region the flow is of strong Couette type. This geometry was considered so as to show the phenomenon of Taylor vortices. When a critical value of Reynolds number is reached Taylor vortices begin to appear in the cylindrical cavity region, this critical value for our geometry was estimated empirically to be Re=58.41 for a Newtonian fluid and lower for a non-Newtonian fluid. The Taylor-vortices are toroidal in shape and rotate in opposite directions when viewed in a meridional plane and appear in contra rotating pairs.

Most of the work carried out on the Taylor-vortex phenomenon has been for infinite length cylinders with no end constraints. However, we have only considered a very short finite length cylinder geometry. Stuart [104] states that for finite length cylinders the boundary conditions which are imposed at the ends will effect the development and pattern of the Talyor-vortices. The boundary condition specified on the top surface of the geometry fig. 5.3 assumes that the surface of the fluid is flat having no normal stress on the free surface and so this condition will exercise a constraint on the flow pattern produced by the rotation of the inner cylinder.

The finite element mesh used to cover the domain is composed of 768 elements and has 1649 nodes, (fig 5.18*).

5.5.2.1 Comparison Of Results From A Program By Bodalia

Results were obtained by adapting a finite difference program by Bodalia [6] for the geometry specified in fig. 5.3 with a vessel height of five. The streamline and rV results at Re=80 for the Newtonian fluid, produced by Bodalia's program run by D.G.Knight are compared with the finite element results. In fig. 5.19* and 5.20* it is seen that both show the appearance of four vortices, whose strength and shape are very similar, with the vortices produced in pairs. Also the plots of rV are similar, (fig. 5.21* and fig 5.22*).

Also presented are streamline plots for the Cross fluid at Re=40, (fig 5.23* and fig. 5.24*). As expected, Taylor vortices are seen to appear at lower Reynolds number than for the Newtonian fluid, due to the shear thinning nature of the fluid giving higher local Reynolds numbers. The streamline plots are remarkably alike, showing an equivalent number of vortices and the rV plots also show good agreement (fig. 5.25* and fig. 5.26*).

5.5.2.2 The Work Of Neitzel [82]

A paper by Neitzel examines the numerical computation of time dependent Taylor-vortex flows in a finite length concentric cylinder geometry, where the motion was initiated by an impulsive start of the inner cylinder from rest. The cylinder used by Neitzel was much longer than used in this study, and as seen from fig 5.27* at Re=62.06 when steady state has been reached Taylor Vortices similar to those achieved here are seen to exist.

5.5.3 Results For The Disc Rotor

Numerical work has already been carried out on the disc stirrer rotating in a cylinder for Newtonian and for pseudoplastic fluids. This is also an axisymmetric problem and the geometry and boundary conditions are defined by fig 5.28 .

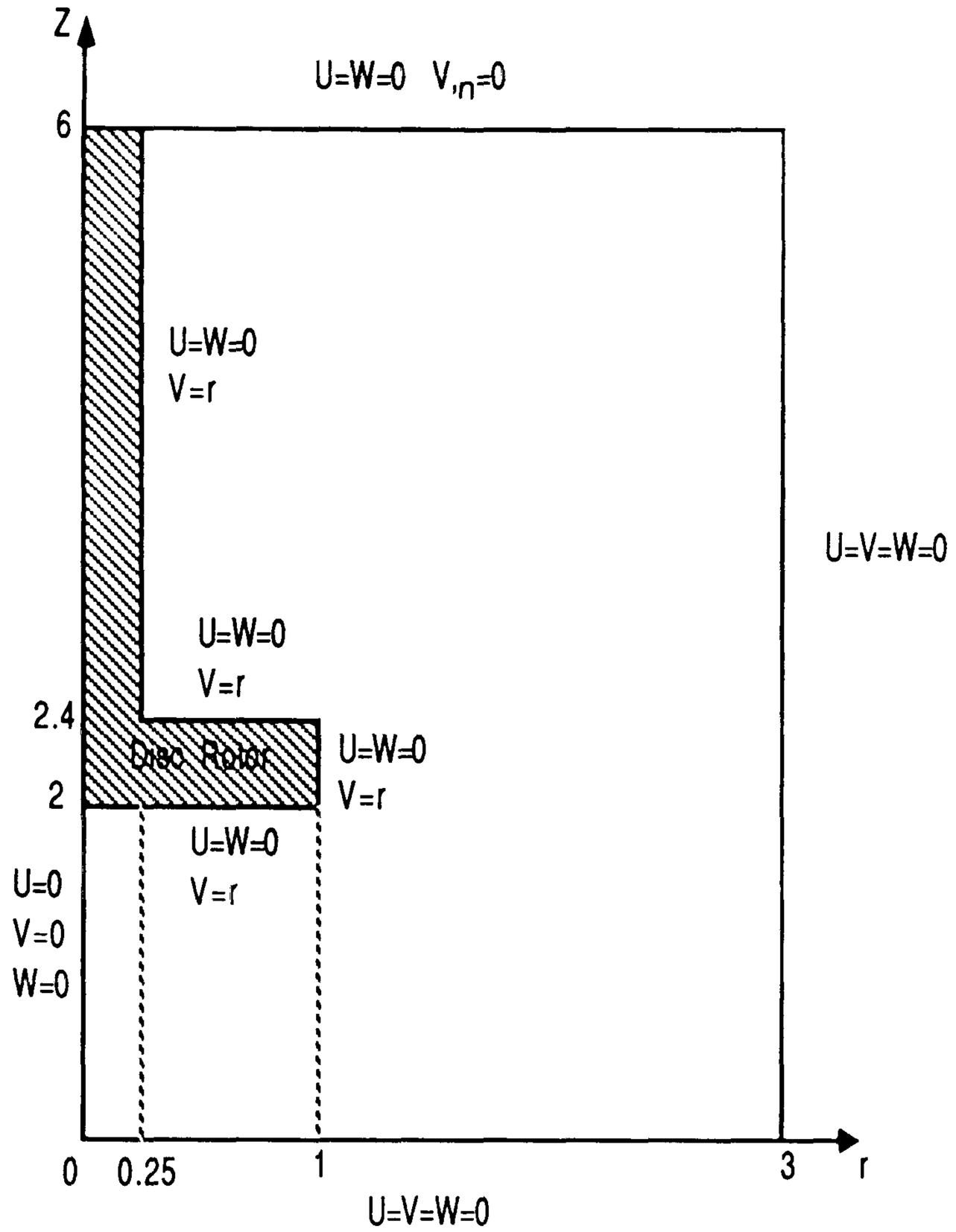


Figure 5.28 Diagram showing disc rotor geometry including boundary conditions.

An important feature of the finite element method is that it is largely unnecessary to modify the finite element program which had been written for the cylindrical rotor in order to make it applicable for the disc rotor. The only changes required are the creation of a new input data file which defines the new geometry and boundary conditions.

The finite element mesh that was used for this problem is made up of 746 elements and 1591 nodes (fig. 5.29*) . The design of the mesh was such that extra refinement was placed close to the disc, as abrupt changes in velocity are expected in this area.

Streamline plots and rV plots have been produced from Re=1 up to Re=500 for the Newtonian fluid. For the case of the Cross fluid plots of streamlines and rV have been obtained from Re=1 up to Re=250.

5.5.3.1 The Work Of Bodalia [6]

Bodalia has carried out work on Newtonian and non-Newtonian flow induced by a rotating disc mounted in a cylindrical vessel with a geometry similar to this work, where the height of her vessel is 5 units and the width is 2.5 units. In her thesis a streamline plot at Re=105 is given (fig 5.30*), and this is compared with our plot at Re=100 (fig 5.31*). Considering the minor differences in the vessel dimensions the streamlines are very alike, and show the same circulatory patterns for equivalent stream function values.

5.5.3.2 The Work Of Spragg et al [103]

Numerical work was carried out by Spragg et al on the theoretical investigation of the steady laminar Newtonian and non-Newtonian flow induced by a rotating disc, mounted coaxially on a shaft in a cylindrical vessel. Bodalia [6] noted that the boundary condition V,r=0 was used by Spragg et al on the axis of symmetry should, in fact, be V=0 as in this and other work [35].Hence Spragg et al's results are not strictly correct, although they do not appear to be greatly affected even near the axis of symmetry.

Spragg et al used a different formula to determine the Reynolds number in his work. Bodalia has determined by comparison Reynolds number Re=105 as equivalent to Spragg et al's Re=67. On comparing the streamline plot at Re=100 with Spragg et al's plot at Re=67 (fig. 5.32*) for the Newtonian fluid (5.33*), it can be seen that the general pattern is the same in both figures with the sign difference in ϕ being due to his definition of the streamfunction.

5.5.3.3 The Work Of Griffiths et al [35]

Griffiths et al made a numerical study of the flows due to a rotating plate or thin disc in a Newtonian fluid and a streamline plot is presented in fig. 5.34*. The general shape and direction of the streamlines are very similar to those obtained in this work, (fig. 5.33*).

The streamlines are seen to be travelling in a clockwise direction below the disc and anticlockwise above it which agrees with us. Griffiths et al also state that their theoretical flow patterns compare very well with experimental work.

5.5.3.4 Results For The Cross Model

For the Cross type fluid we present streamlines at Re=200 (fig. 5.35*) and this is compared with a plot at Re=200 for a Newtonian fluid (fig. 5.36*). The streamlines are expected to be a lot closer to the disc in the non-Newtonian case and the circulatory pattern is similar to a flow at a higher Reynolds number for the Newtonian fluid. This is due to the shear thinning nature of the fluid.

5.5.4 Results For The Axisymmetric Rushton Turbine.

Investigational work has been performed on the axisymmetric "Rushton turbine" geometry, also known as the flat blade paddle geometry, for a Newtonian fluid. This again is a 3-D axisymmetric problem and the geometry and boundary conditions are defined by fig. 5.37.

The finite element mesh that was used for this problem is made up of 196 elements and 453 nodes (fig. 5.38*) and was fairly coarse . The design of the mesh was such that extra refinement was placed in the vicinity of the flat blade paddle. Streamline and rV plots are presented for Re=1 (fig. 5.39* and fig. 5.40*) and for Re=100 (fig. 5.41* and fig. 5.42*). Solving for this geometry shows the versatility of the finite element method over the finite difference method as it would be difficult in finite differences to deal with this geometry.

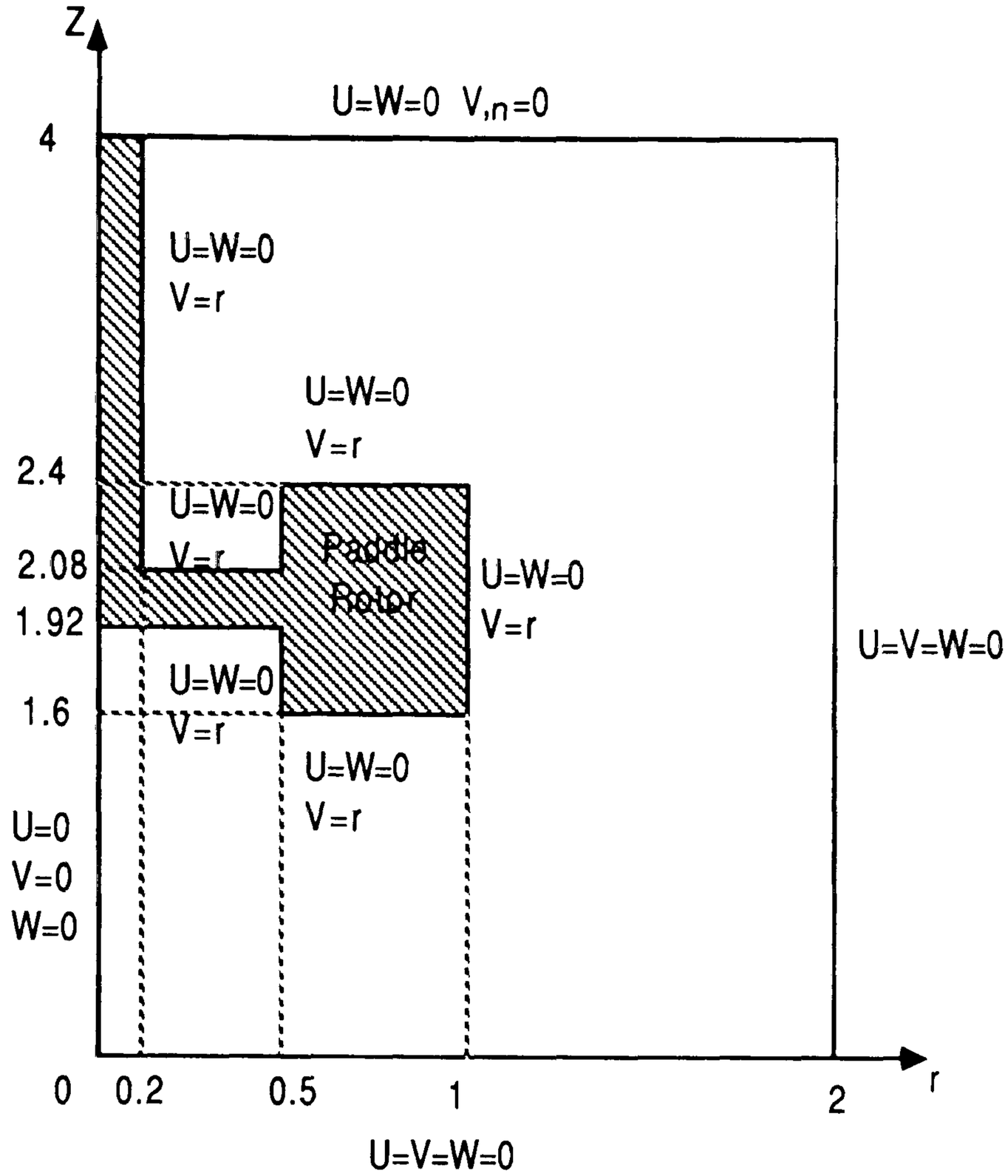


Figure 5.37 Diagram showing flat blade paddle geometry

including boundary conditions.

5.5.4.1 <u>The Work Of Voncken et al [111]</u>

It was only possible to make a rough comparison with the work of Voncken et al as they had used a much larger arrangement than ours and the streamline plot presented was for a faster flow; however the results were broadly similar with ϕ contours.

5.6 <u>Time Dependent Steady Flows</u> - <u>Applied to Colour Band Mixing</u>

The aim of this section is to investigate colour band and dispersive mixing for the geometries described in the preceding section of this chapter (fig.5.3 with A=2 and fig.5.28*). This can be achieved by solving the three-dimensional axisymmetric convection/diffusion equation (5.33) given in r,θ,z coordinates.

$$C_{,t} - D/r[(rC_{,r})_{,r} + (rC_{,z})_{,z}] + (UC_{,r} + WC_{,z}) = 0 \quad , \qquad (5.33)$$

where C signifies the concentration, D the diffusion coefficient (4.36) with U and W the velocities in the r and z directions. A band of colour is added to the flow when steady state has been reached. From the work done on concentration in chapter 4 section 10 two formulations of the problem are required; firstly the semidiscrete approach is employed to formulate the problem, and secondly an upwinded version of the first formulation is obtained. The details concerning these two approaches are outlined in chapter 4 section 10. In the first formulation the Galerkin method is adopted for the spatial approximation and the Crank/Nicolson finite difference method for the temporal approximation. As mentioned earlier for axisymmetric flow, the integration is performed over a three-dimensional domain surrounding the axis of symmetry and so it is necessary to employ (5.13). An approximation of the form (5.34) is used

$$C(t,r,z) = \sum_{j=1}^{m} S_j(r,z)C_j(t) \quad , \qquad (5.34)$$

where $S_j(r,z)$ is the spatial term expressed as S and $C_j(t)$ is the temporal term expressed as C throughout the rest of this formulation. The first stage of the formulation is to apply the Galerkin method to the spatial terms in the equation (5.33). This gives

$$\iint (C_{,t} - D/r[\, C_{,r} + rC_{,rr} + rC_{,zz}\,] + UC_{,r} + WC_{,z})2\pi rS_i \, drdz = 0 \quad . \qquad (5.35)$$

Applying the divergence theorem to the second order terms gives:

$$\iint D(C_{,rr} + C_{,zz})2\pi rS \, drdz = -\iint 2\pi D(C_{,r}S + C_{,r}rS_{,r} + C_{,z}rS_{,z})drdz + 2\pi D\int rSC_{,n}\, ds = 0 \quad . \qquad (5.36)$$

Finally substituting (5.34) and (5.36) into (5.35) and adopting tensor product notation yields

$$C_{j,t}\langle S_j;rS_i\rangle + C_j[\, D(\langle S_{j,r};rS_{i,r}\rangle + \langle S_{j,z};rS_{i,z}\rangle) + (U\langle S_{j,r};rS_i\rangle + W\langle S_{j,z};rS_i\rangle)\,] - D\int rS_iC_{j,n}\, ds = 0 \quad , \qquad (5.37)$$

where to simplfy the expression we have omitted the 2π term. This is written in matrix form as

$$[A]\{C_{,t}\} + [B]\{C\} = 0 \quad . \qquad (5.38)$$

The formulation of the temporal part is identical to that described in chapter 4 section 10 and is thus not described again here.

The second formulation involves incorporating the SU/PG method into the first formulation. For the three dimensional axisymmetric case the upwind perturbation is added in the direction of the streamlines which only involve U and W velocities and not V velocities as this is not a fully three dimensional formulation i.e. we are dealing with two dimensional elements, therefore V is excluded from the upwind contribution. The method of formulating this approach is therefore very similar to the two dimensional case described in great detail in chapter 4 section 10 where the only real difference is that the V term in the 2-D case is replaced by the W term in the 3-D case, and it is thought unnecessary to repeat the description of the formulation here.

5.6.1 <u>The cylinder geometry</u>

The dimensions of the geometry and initial conditions are shown in fig.5.43. The boundary conditions specified on the whole of the boundary are homogeneous Neumann conditions.

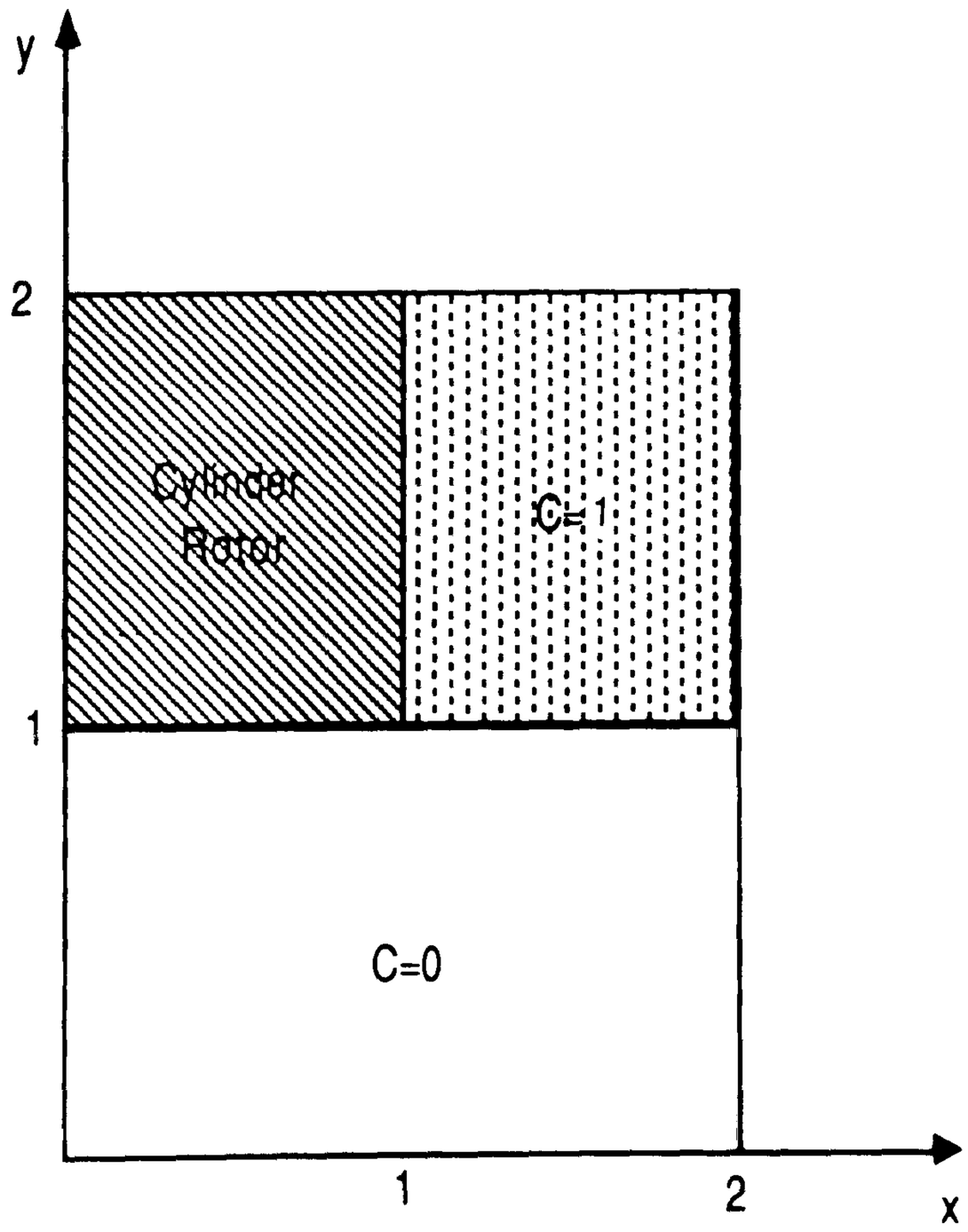


Figure 5.43

The total concentration volume in the region is given by

$$\text{concentration volume} = \frac{\iint C\, 2\pi r\, drdz}{\iint 2\pi r\, drdz} = \text{const.} \qquad (5.39)$$

During mixing the overall volume concentration of colour in the region remains constant, that is equal to its initial value, and so can be used as a check on how well the solution is behaving. The finite element mesh data, represented by (fig.5.5*), which is composed of six noded triangular elements is transformed into data for a mesh of four noded quadrliateral elements.

In testing the non upwinded (NU) program it is possible to obtain good results for Re=100. In figures 5.44a*, 5.44b*, 5.44c*, 5.44d*, 5.44e*, 5.44f* and 5.44g* respective plots are presented showing the evolution of the concentration field after 10s, 50s, 100s, 133.4s (which is a 25% mix), 160.0s (which is a 50% mix), 195.4s (which is a 75% mix) and 432.6s (which is a 95% mix). The NU program for Re=400 produces spurious wiggles in its solutions after only a few time steps, however the SU/PG program produces excellent results by smoothing out the spurious parts of the solution.

It was decided to further investigate how well the SU/PG program would work when the solutions obtained from the NU program had totally broken down and become meaningless. For Re=800 the NU program produces spurious wiggles which rapidly degrade the solution as it proceeds in time and the values obtained at different time steps for the volume of concentration were found to be oscillating wildly, however on using the SU/PG program the results remained consistent, smoothing out the solution without over damping. This is shown in the comparative plots of the solution produced at different time steps by the NU program and the SU/PG program. Fig.5.45a* shows a plot of a non-upwinded solution and fig.5.45b* shows a plot of an upwinded solution both at t=10s for Re=800. Plots are also presented at t=50s and t=100s for Re=800 for both non-upwinded solutions, fig.5.46a* and fig5.47a* respectively, and upwinded solutions, fig.5.46b* and fig.5.47b* respecitvely.

D.G. Knight(unpublished) has implemented a finite difference program for the same geometry. The solutions compared well at t=20s for Re=10 as shown in fig5.48a* (Knight) and fig.5.48b*, and for a 75% mix at t=231.8s in fig.5.49a*(Knight) and at t=235.6s in fig.5.49b*. At Re=100 the results produced by Knight using a non-upwinded program show many spurious wiggles, fig.5.50a*. On employing an upwinding strategy it became possible for Knight to obtain wiggle free solutions, but on viewing a plot of the solution at t=10s (fig.5.50b*) it can be seen that to much artificial diffusion is present compared with fig.5.44a*.

5.6.2 The Disk geometry

The geometry and initial conditions for the disk rotor problem are shown in fig.5.51, with the boundary conditions being homogeneous Neumann conditions.

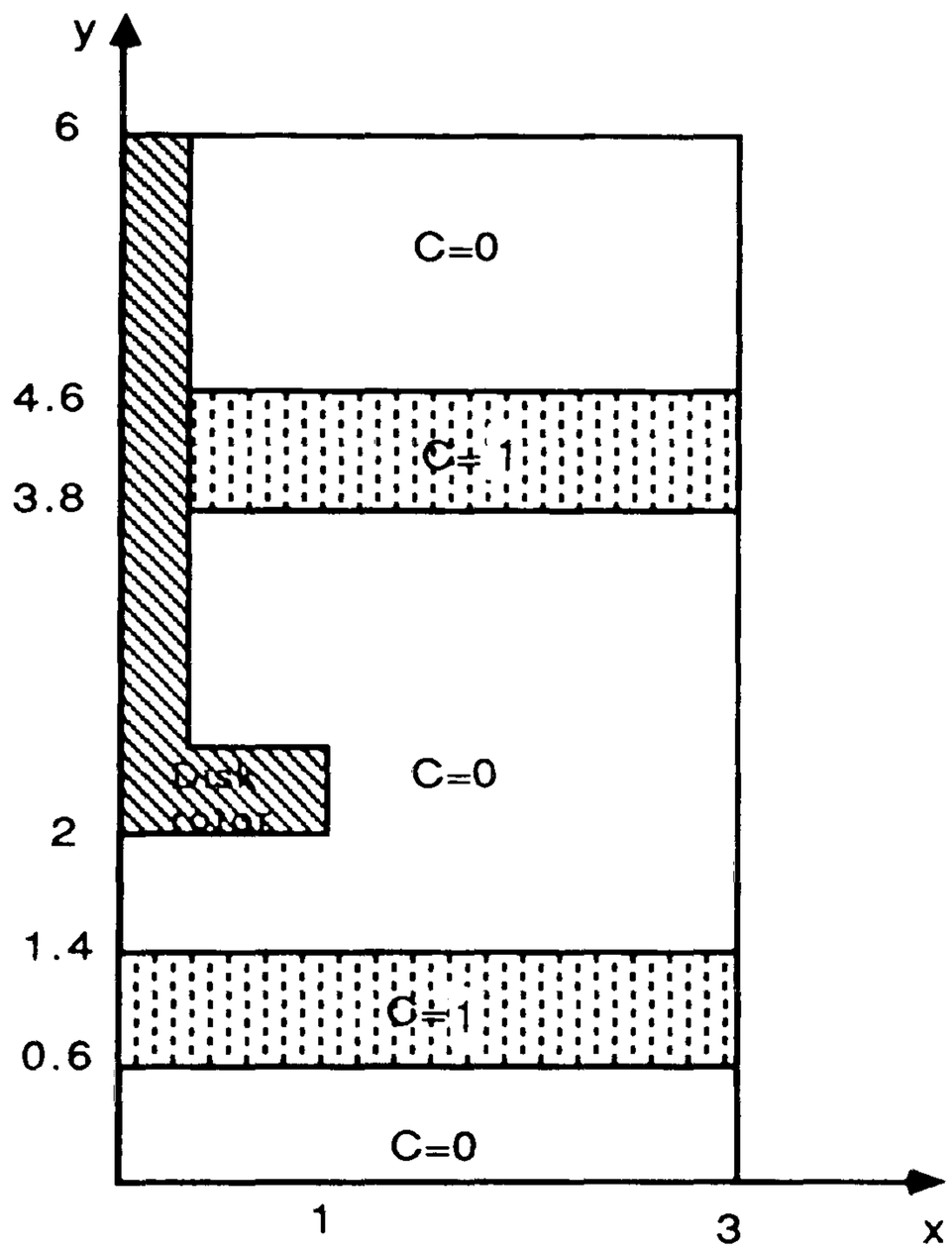


Figure 5.51

5.6.2.1 The work of Spragg et al [103]

The geometry and initial conditions used by Spragg et al are similar to those in this report (fig.5.52*), but as has been mentioned in an earlier section Spragg et al employ a different formula to determine Reynolds number. They gave results for Re=17 which is equivalent

to our Re=26.7. Their results are for a non-Newtonian fluid which is represented by the following model,

$$\eta = 0.2 + \frac{0.8}{(1 + q^2)^{0.25}} \quad . \tag{5.39}$$

We have used the same model in our calculations to enable us to make a comparison with the work of Spragg et al. These solutions are at 5 revs (fig.5.53a*) and 20 revs (fig5.54a*) and are equivalent to t=31.4s (fig5.53b*) and t=125.7s (fig.5.54b*). On comparing the plots it is seen that there is a broad similarity between them.

5.7 **Summary**

The transition from the 2-D case to the 3-D axisymmetric case was accomplished with little difficulty. The U,V,P approach was extended to the U,V,W,P approach for the 3-D axisymmetric case. Three separate mixing geometries were investigated for Newtonian and Cross type fluids:

(a) The cylinder rotor geometry was considered for two different size rotors and vessels. In the first instance a short rotor was used and results up to Re=1000 were obtained for the Newtonian fluid. The results compared well with those exhibited by other workers employing finite difference methods and for the higher Reynolds numbers we were able to show the occurrence of axisymmetric vortex breakdown. Then a tall rotor geometry was considered in order to show the phenomenon of Taylor vortices for both types of fluid, this was achieved and the results obtained compared well with results produced using finite difference methods.

(b) The disc rotor problem was considered next as it was a slightly more complex geometry. The results produced were again in excellent agreement with the work of others.

(c) Finally the flat blade paddle rotor problem was considered using a fairly coarse mesh and the results obtained were satisfactory.

These three problems were solved by the same finite element program as developed in this chapter without having to make any modification to the actual program. The mesh data for each geometry was produced by the mesh generator program in the correct format for the finite element program.

For the cylinder and disk geometries some investigational work was undertaken into colour band mixing for time dependent steady flows. A semidiscrete finite element/finite difference formulation was employed and good results were obtained for the cylinder geometry at medium values of Reynolds numbers, i.e. up to Re=100. It was necessary to introduce upwinding into the formulation inorder to continue obtaining excellent results for significantly higher Re values, i.e. up to Re=800. Dispersive mixing of the disk rotor problem for the Cross fluid was also investigated and our results compared well with those of Spragg et al.

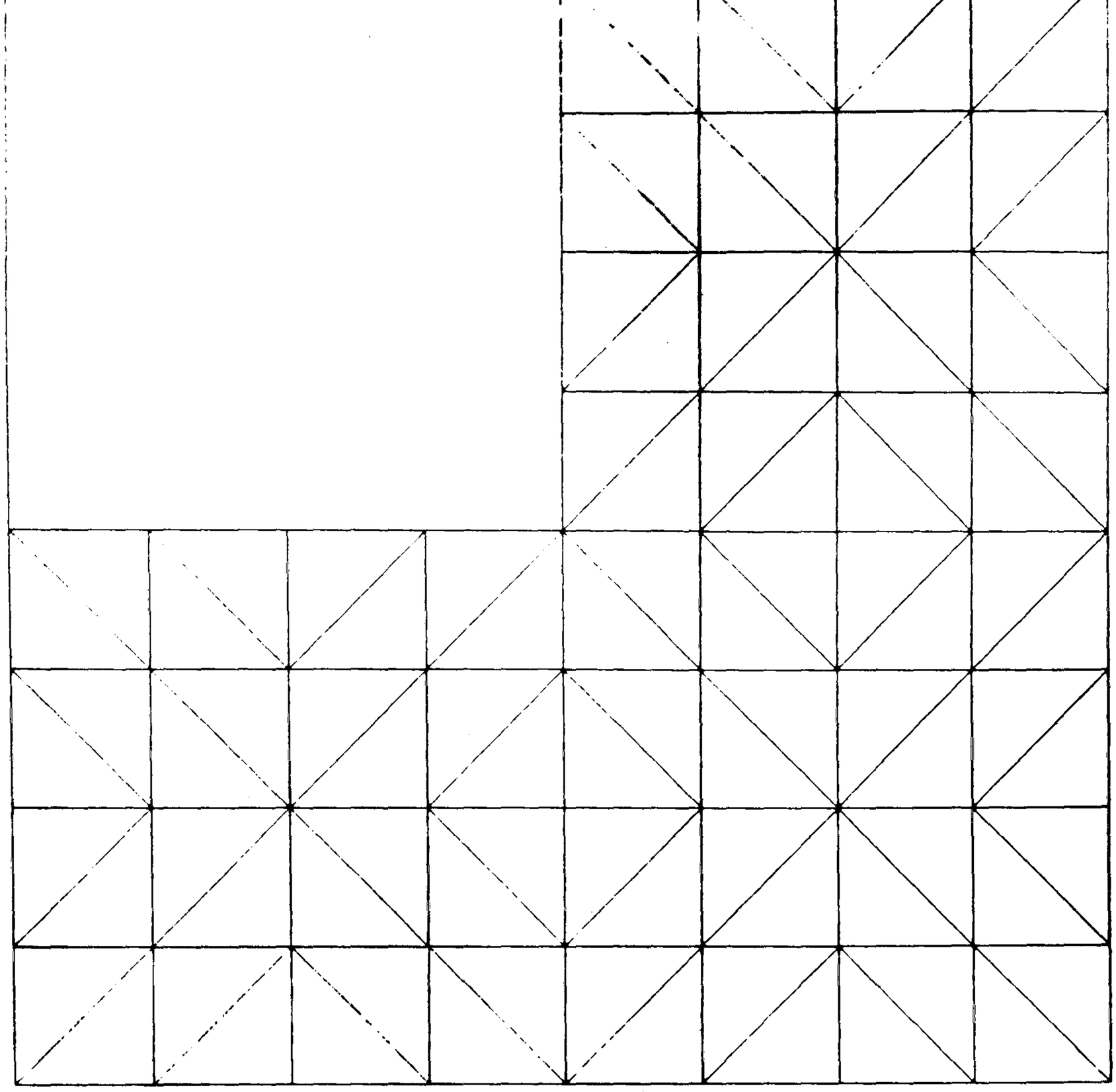


Figure 5.4 Coarse mesh.

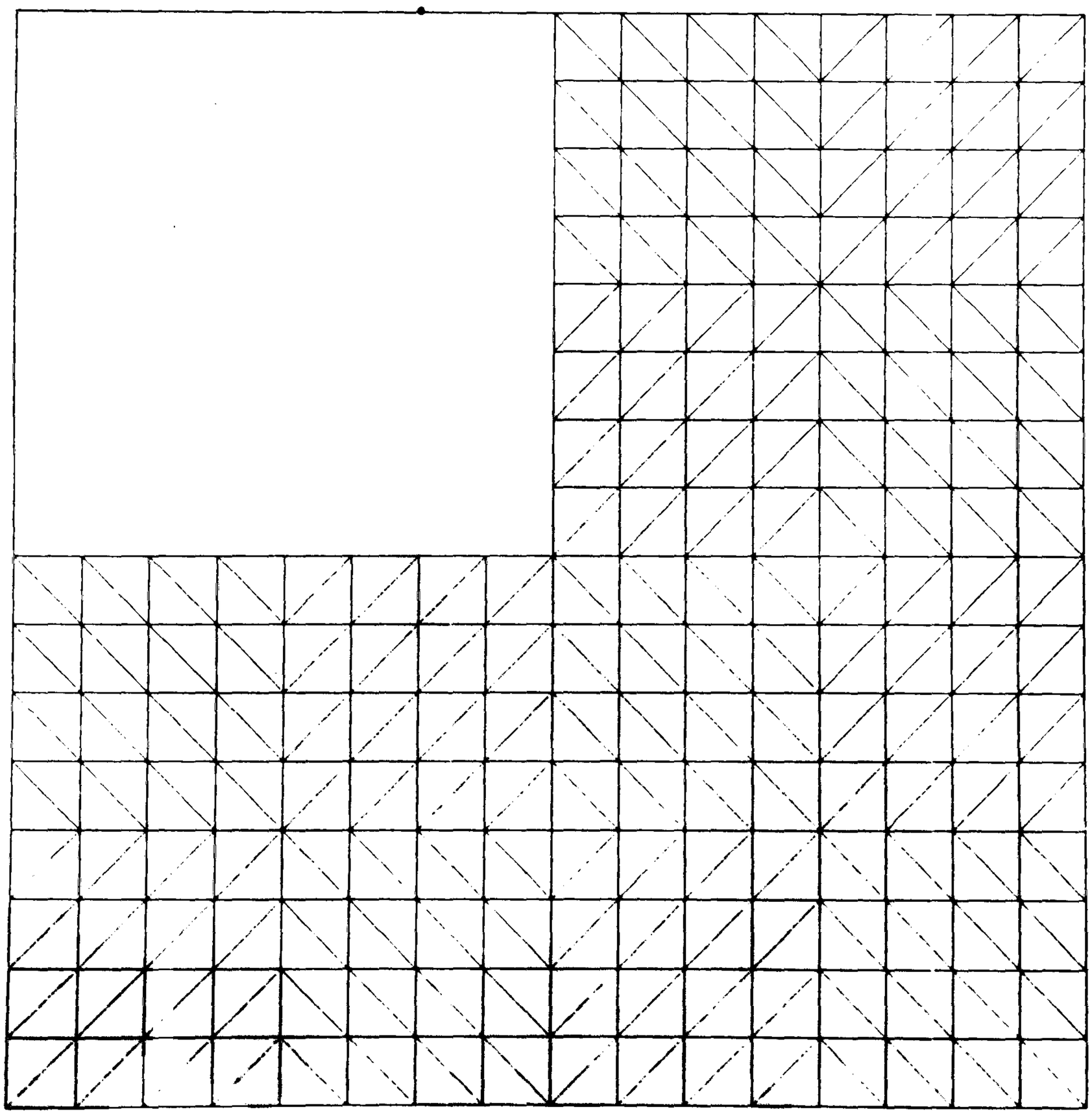


Figure 5.5 Refined mesh.

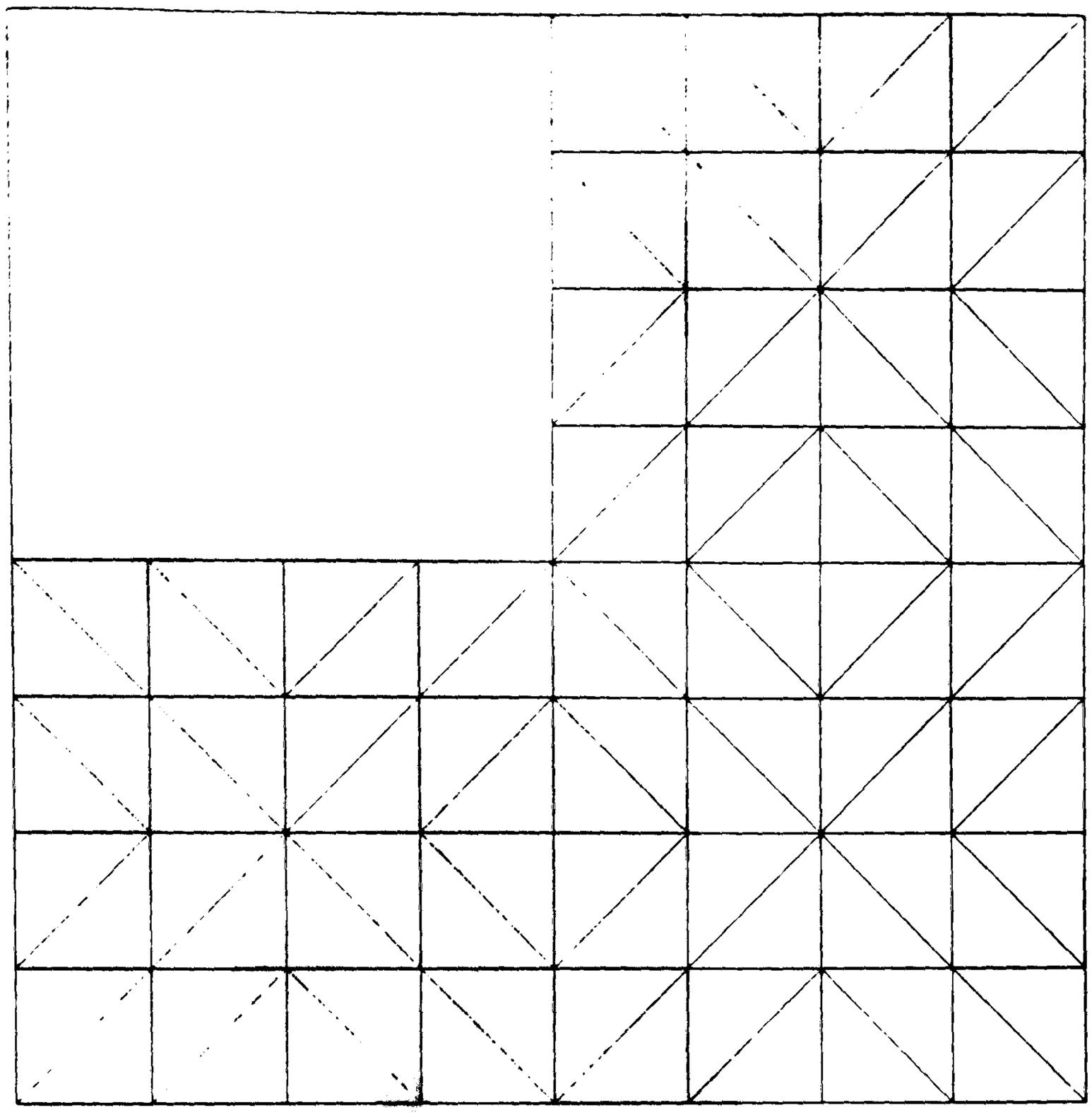


Figure 5.4 Coarse mesh.

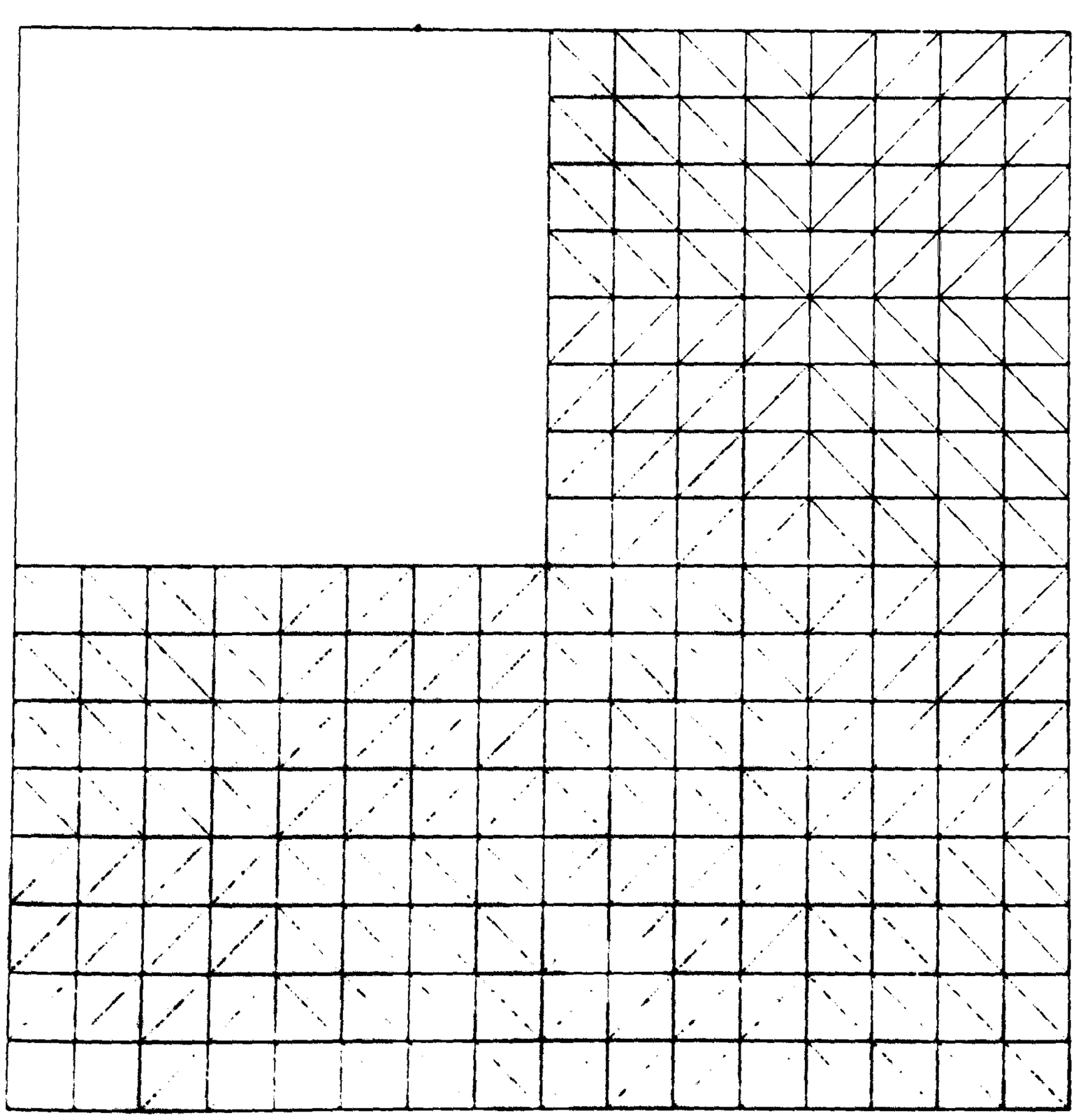
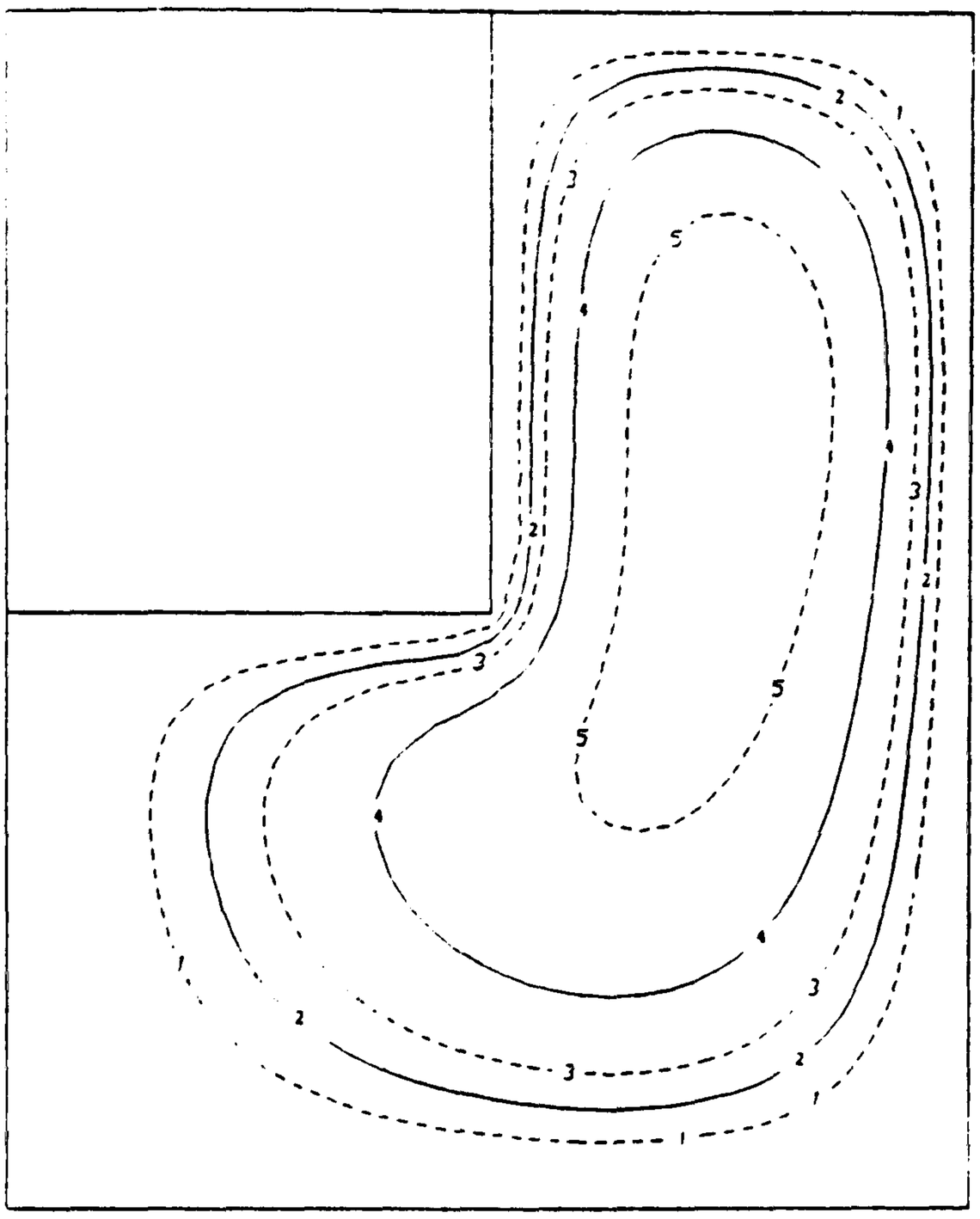


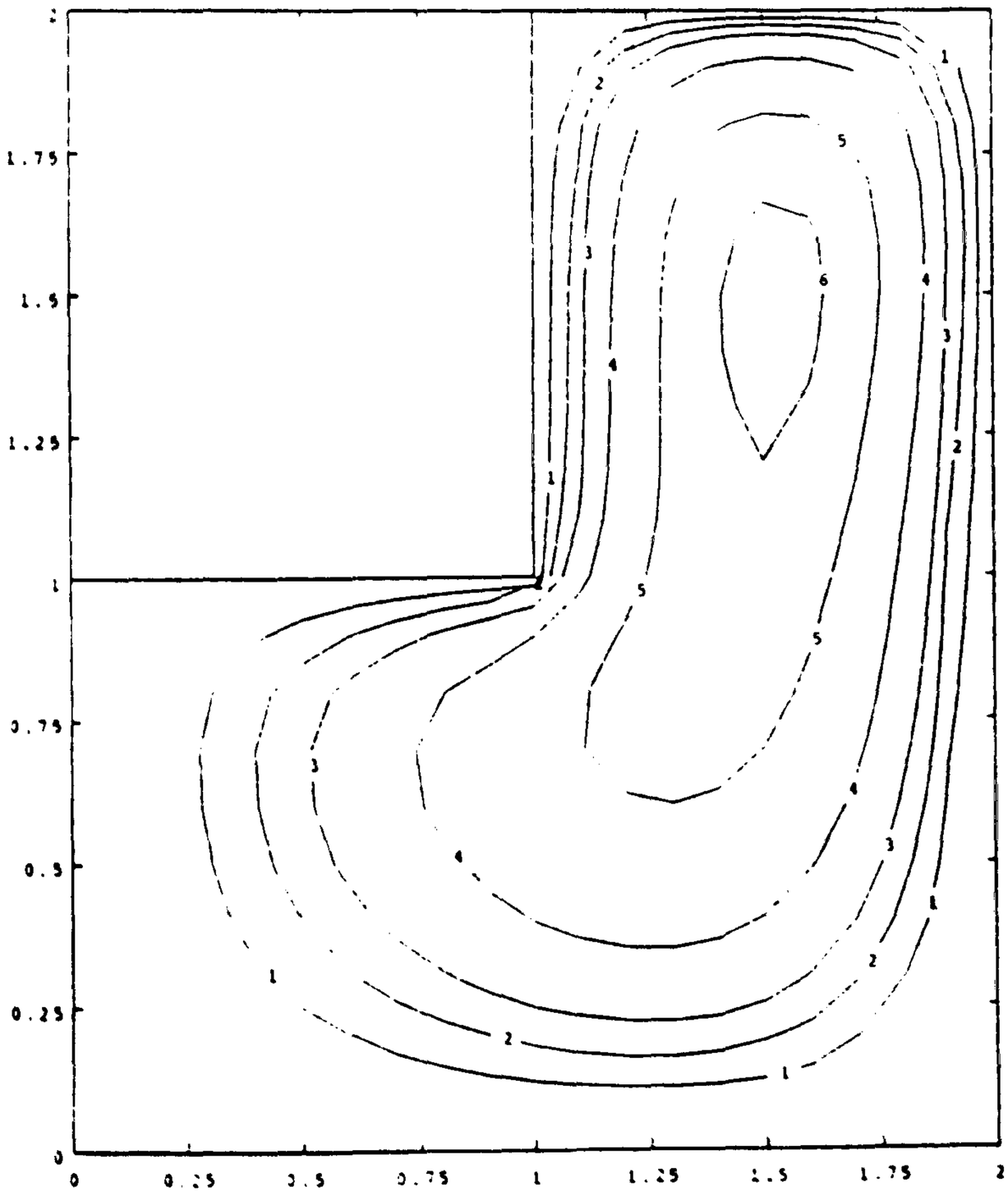
Figure 5.5 Refined mesh.



CONTOUR KEY	
1	-0.300E-02
2	-0.600E-02
3	-0.100E-01
4	-0.200E-01
5	-0.400E-01
6	-0.600E-01

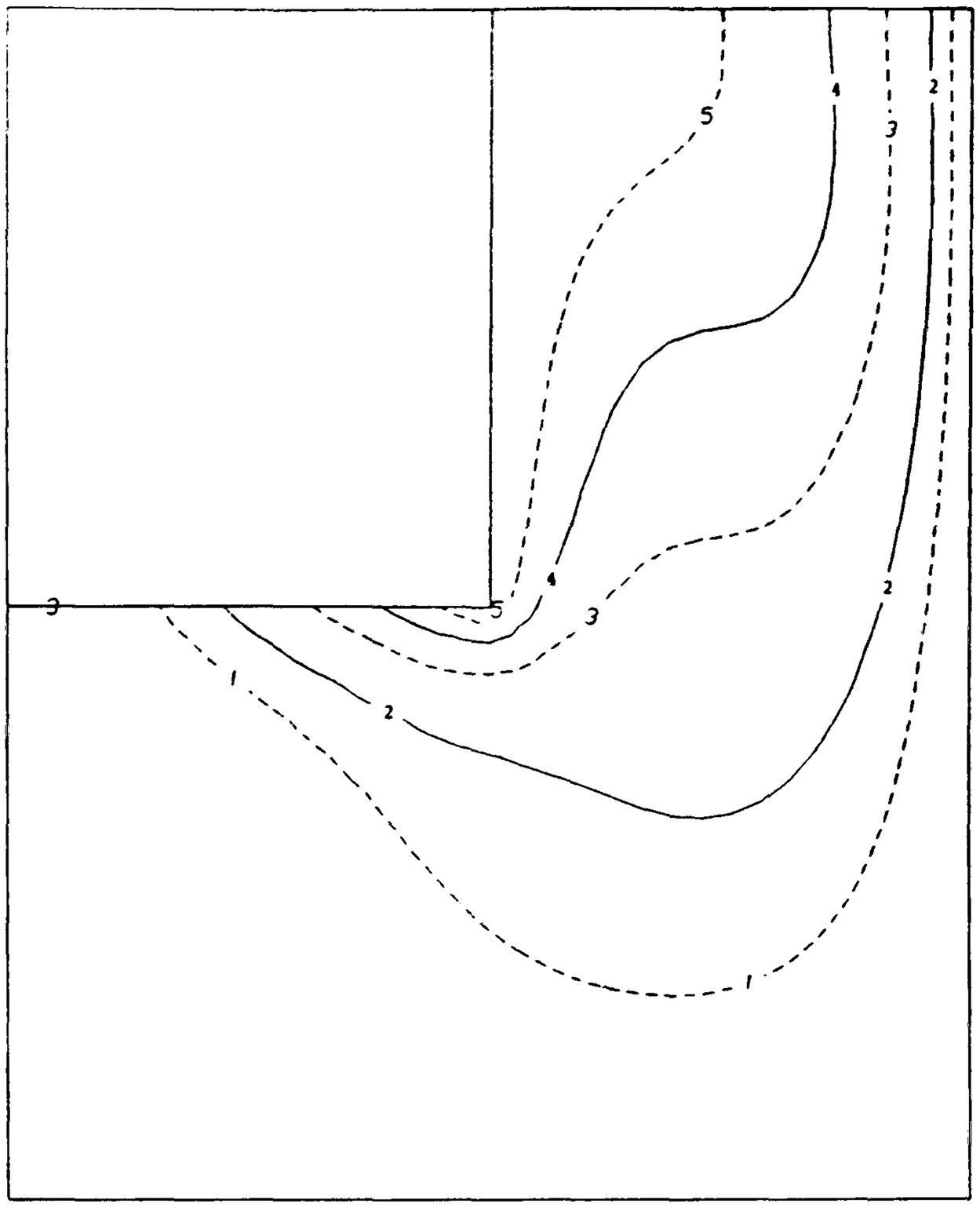
CONTOUR KEY	
1	-0.003
2	-0.006
3	-0.010
4	-0.020
5	-0.040
6	-0.060

Figure 5.7 Streamlines at Re=100 for a Newtonian fluid.



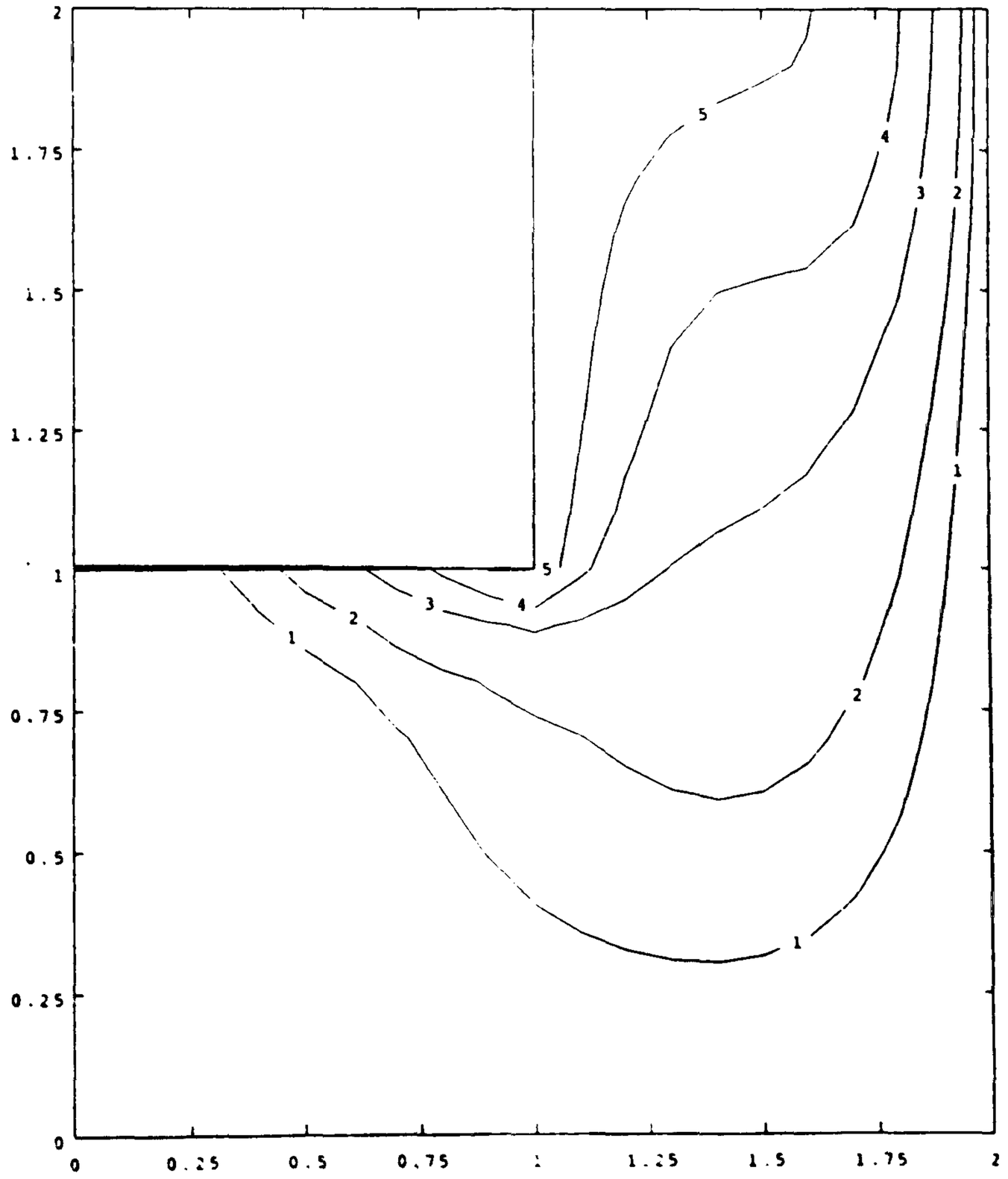
CONTOUR KEY	
1	-0.003
2	-0.006
3	-0.010
4	-0.020
5	-0.040
6	-0.060

Figure 5.8 Streamlines at Re=100 for a Newtonian fluid by Bodalia.



CONTOUR KEY	
1	0.100E+00
2	0.200E+00
3	0.400E+00
4	0.600E+00
5	0.800E+00

Figure 5.9 Plot of rV contours at Re=100 for a Newtonian fluid.



CONTOUR KEY	
1	0.10
2	0.20
3	0.40
4	0.60
5	0.80

Figure 5.10 Plot of rV contours at Re=100 for a Newtonian fluid by Bodalia.

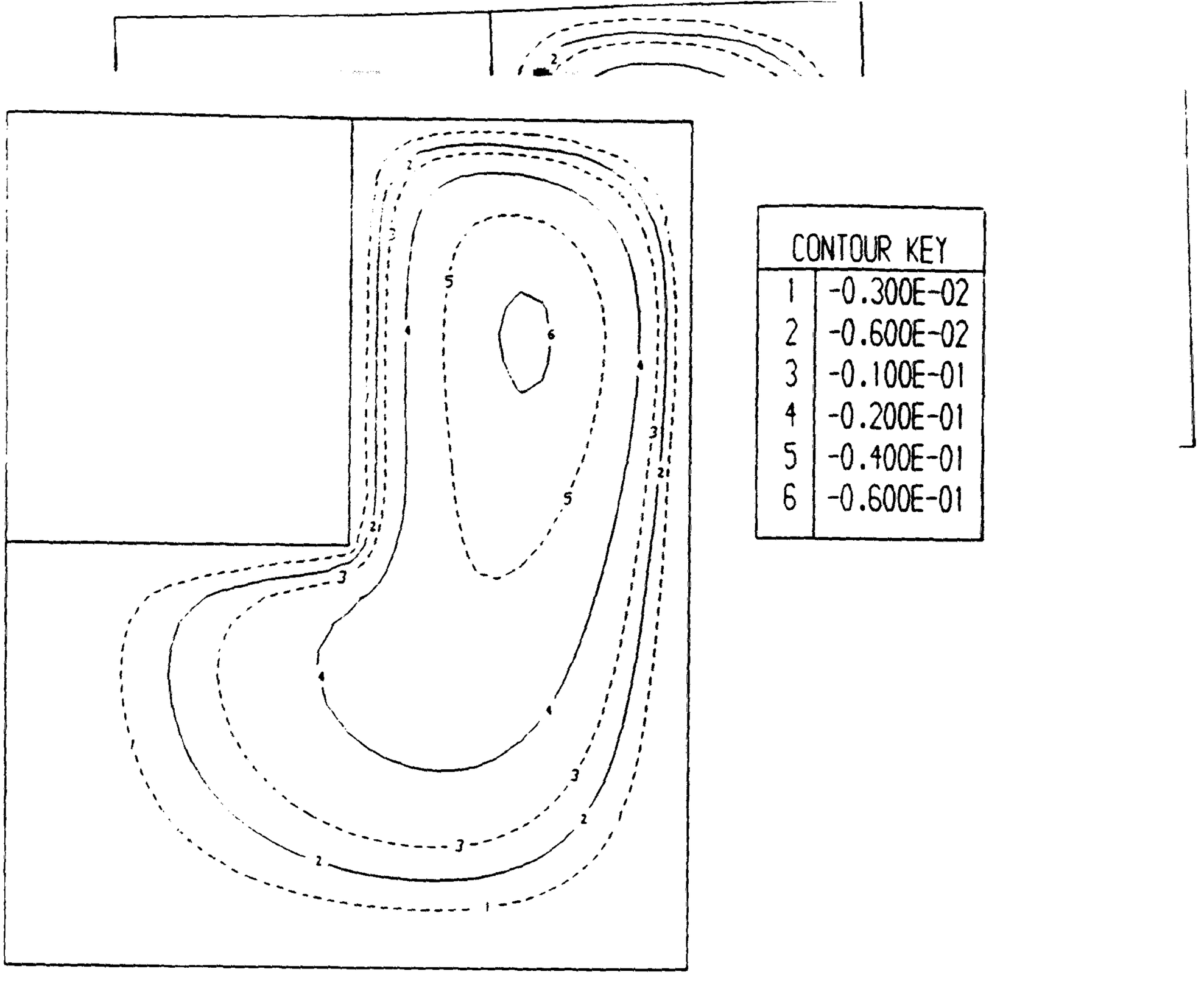


Figure 5.11 Streamlines at Re=100 for a variable viscosity fluid.

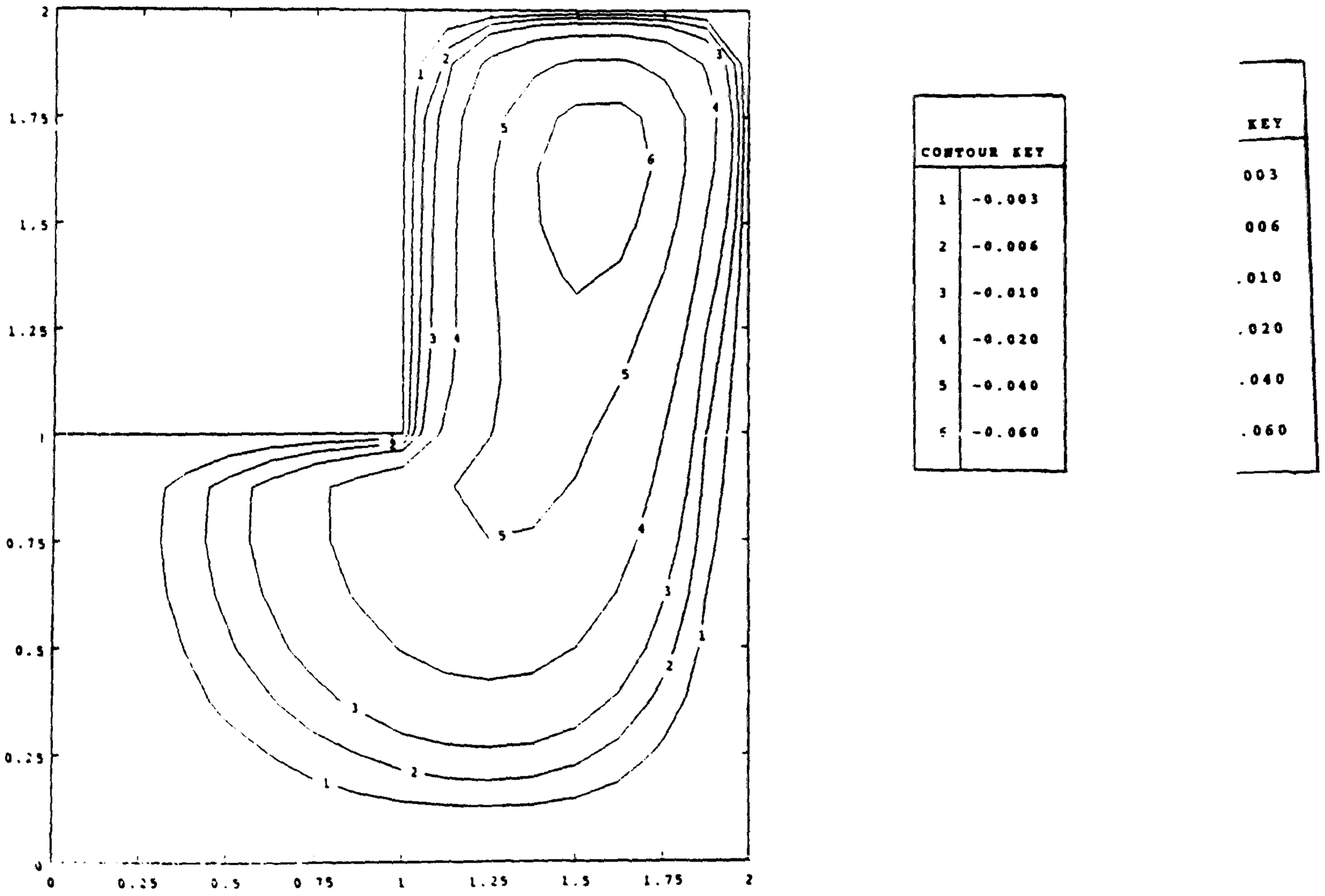


Figure 5.12 Streamlines at Re=100 for a variable viscosity fluid by Bodalia.

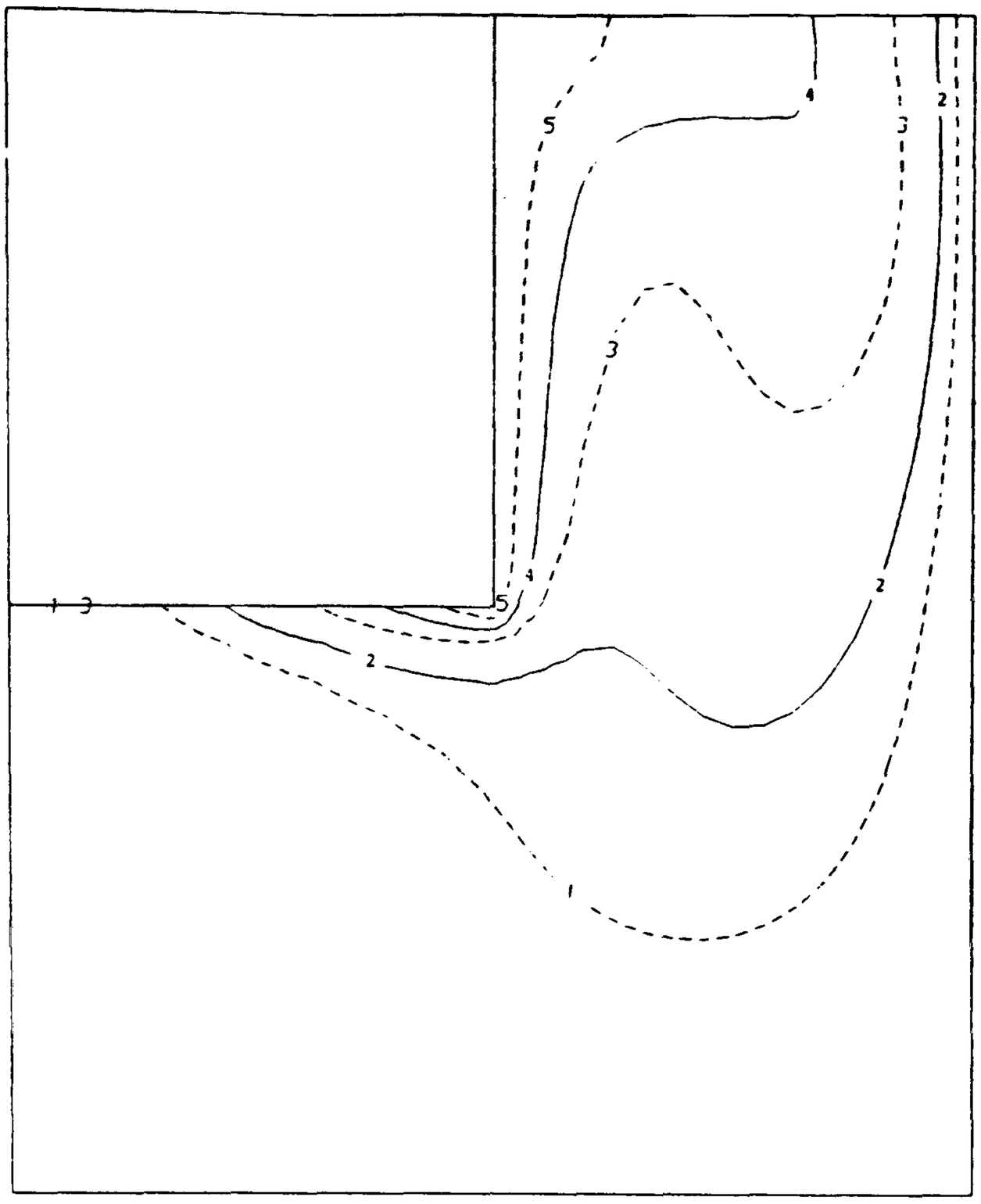


Figure 5.13 Plot of rV contours at Re=100 for a variable viscosity fluid.

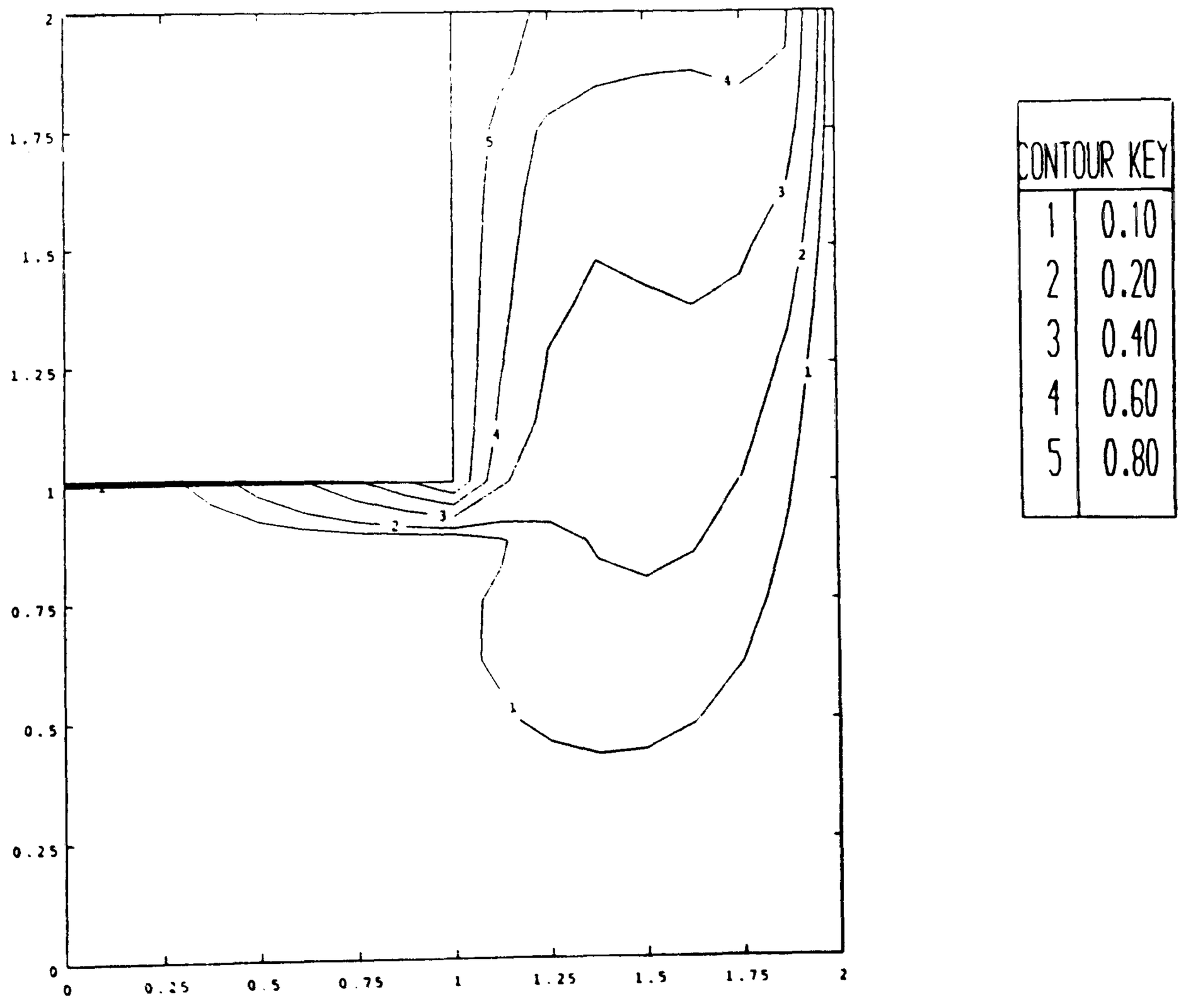


Figure 5.14 Plot of rV contours at Re=100 for a variable viscosity fluid by Bodalia.

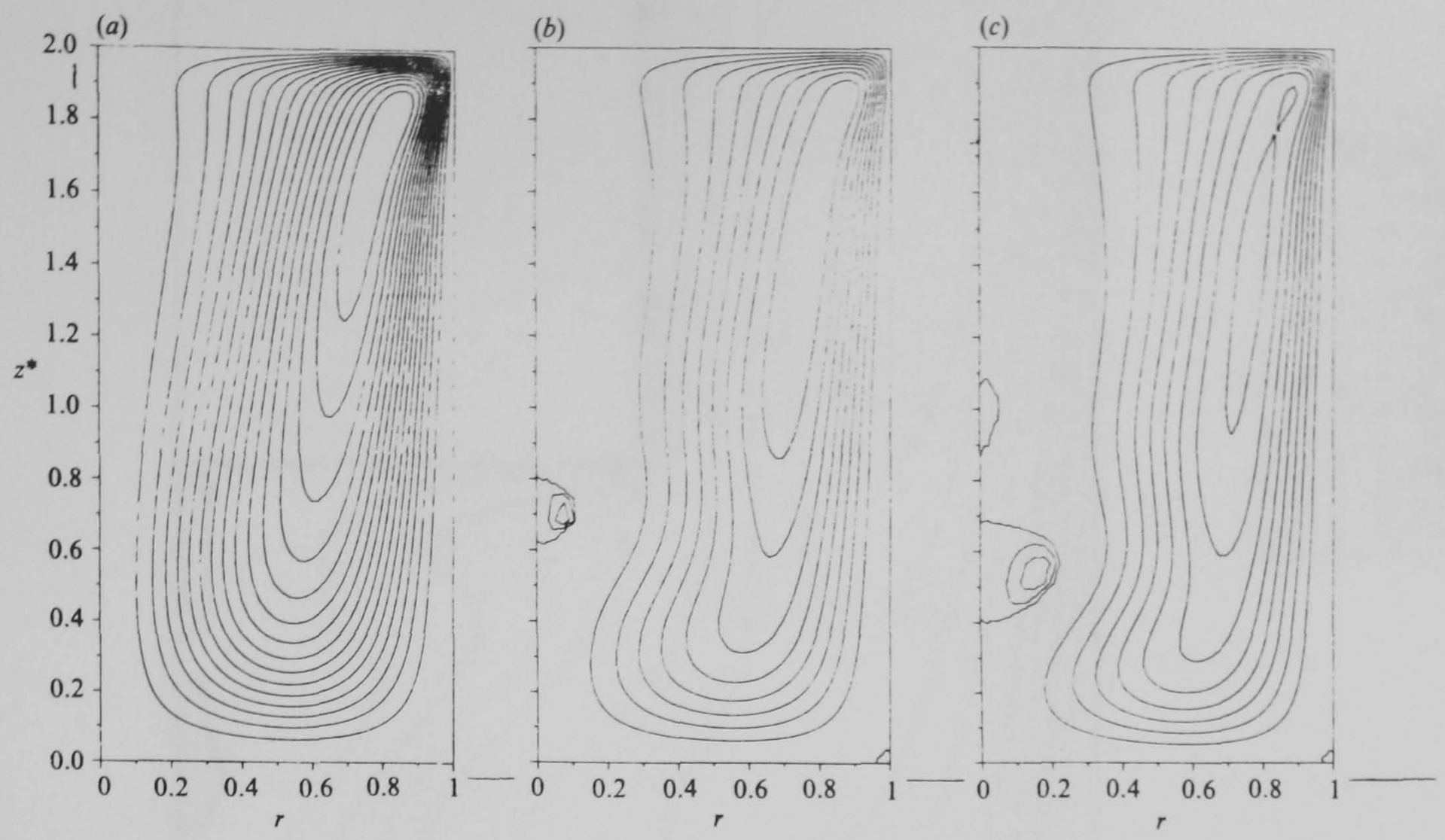
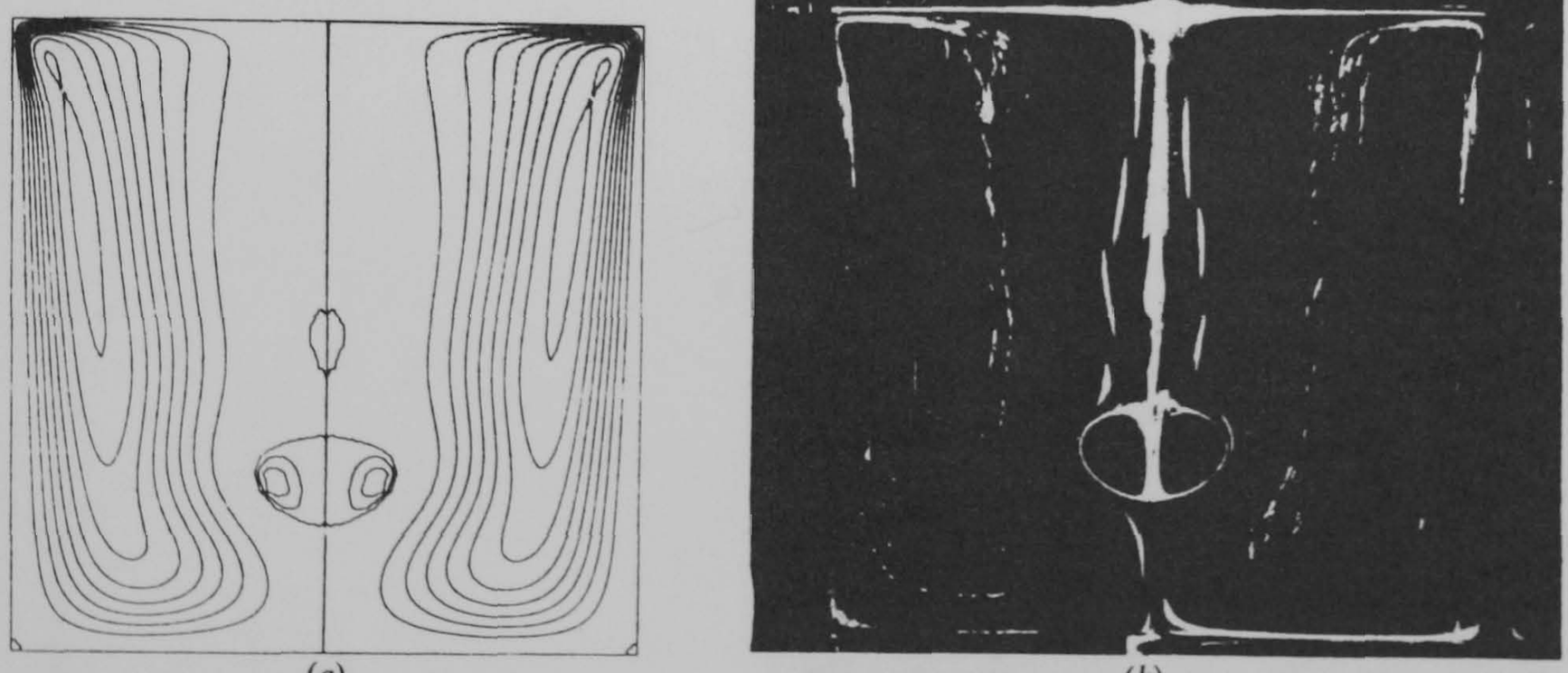


Figure 5.15 Streamlines at steady state for d=2 and for (a) Re=1002, (b) 1492, and (c) 1854 by Lugt and Abboud.



(a) (b)

Figure 5.16 Streamlines for d=2 and for Re=1854 (a) Numerical computation and (b) photograph by Escudier.

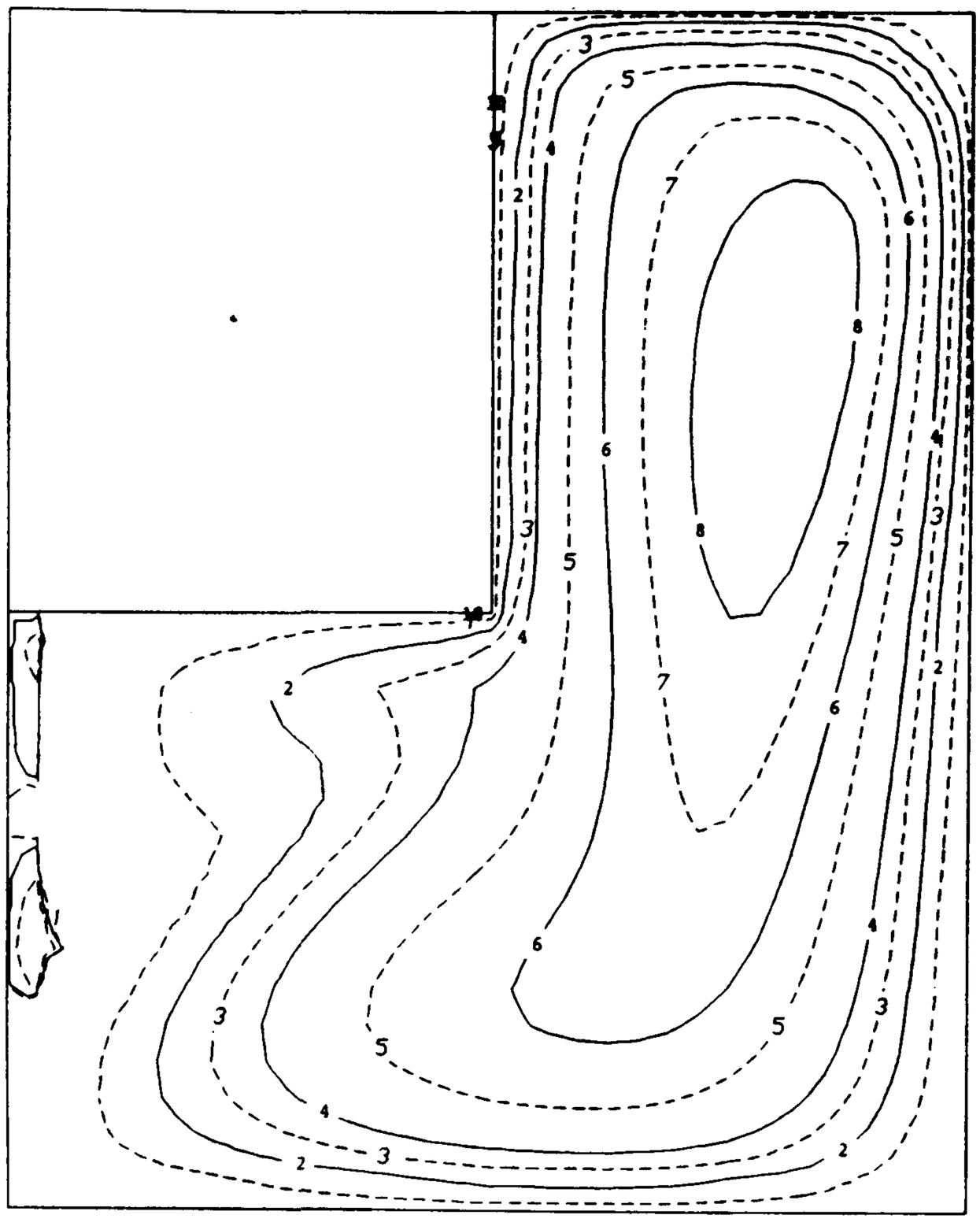


Figure 5.17 Streamlines at Re=1000 for a Newtonian fluid.

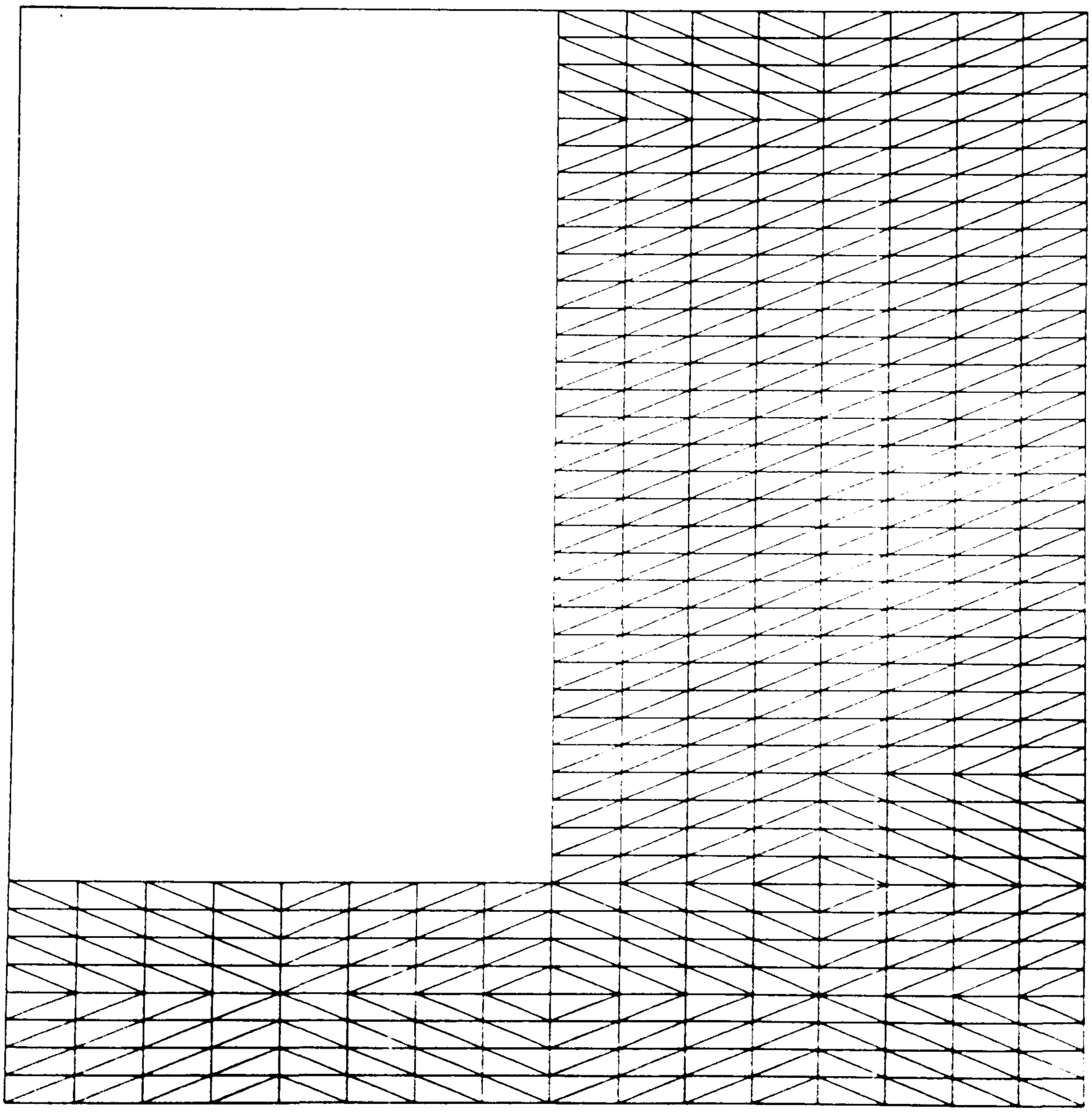
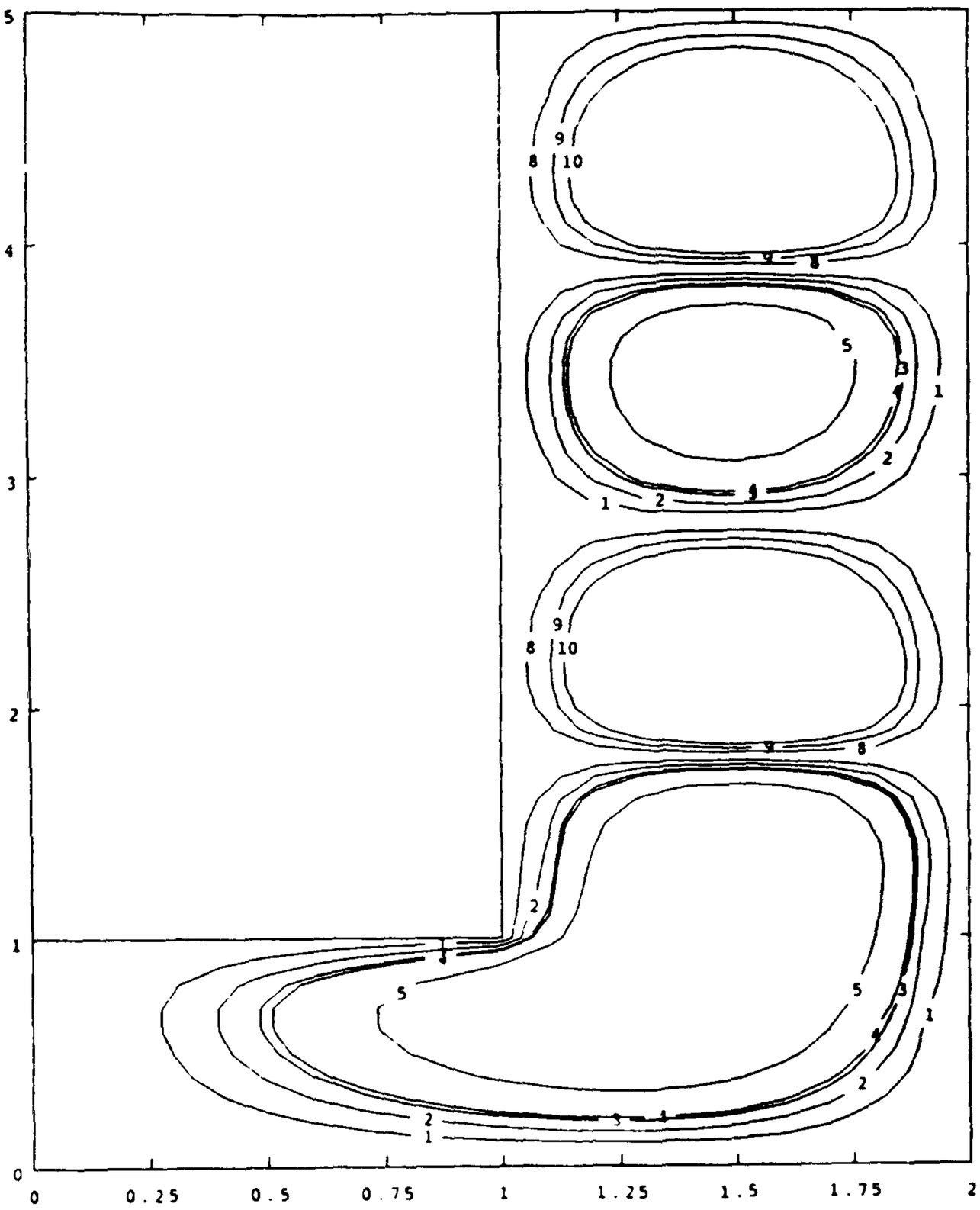
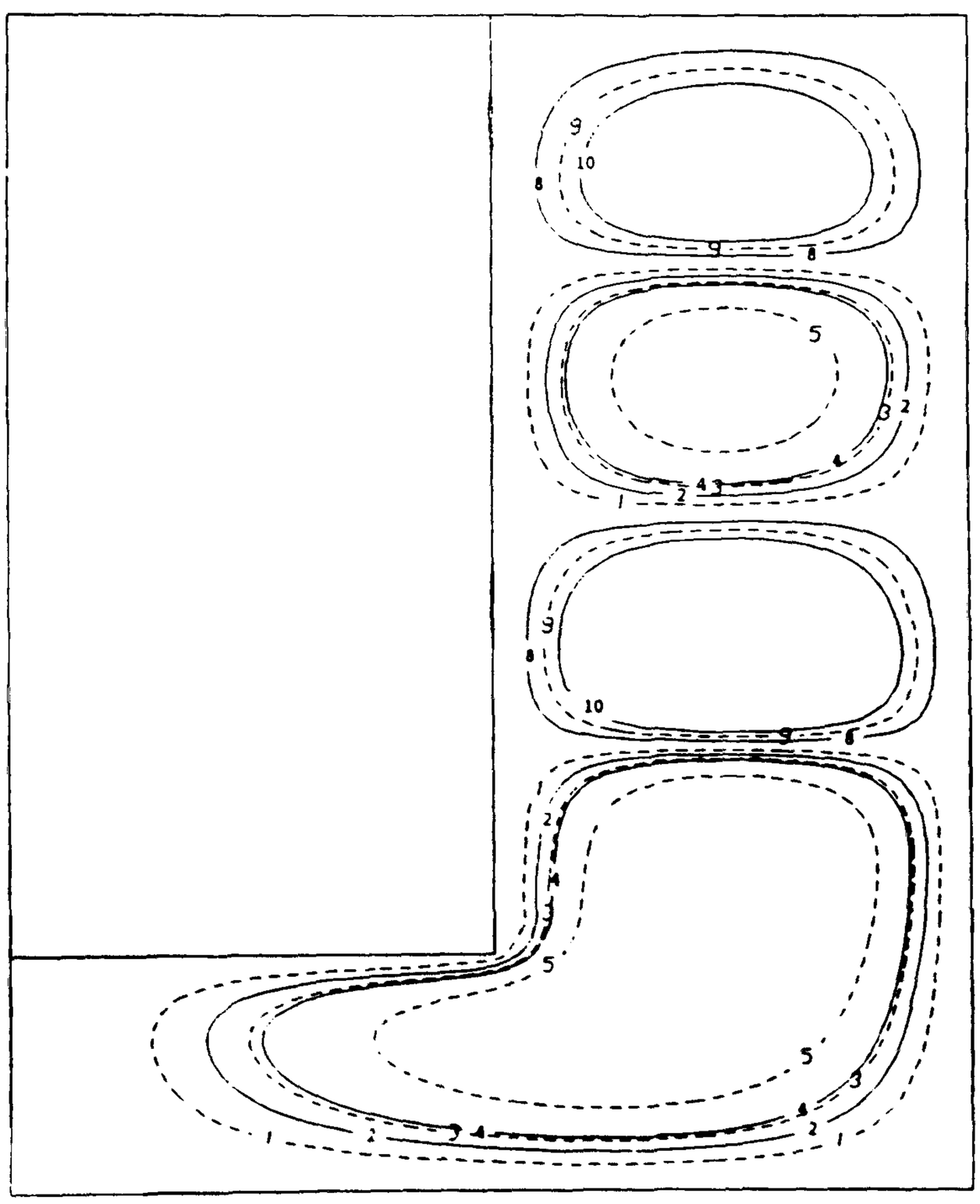


Figure 5.18 Finite element mesh.



CONTOUR KEY	
1	-0.003
2	-0.006
3	-0.009
4	-0.010
5	-0.020
6	-0.070
7	0.070
8	0.003
9	0.006
10	0.009

Figure 5.19 Streamlines at Re=80 for a Newtonian fluid produced by Bodalia's program.



CONTOUR KEY	
1	-0.300E-02
2	-0.600E-02
3	-0.900E-02
4	-0.100E-01
5	-0.200E-01
6	-0.700E-01
7	0.700E-01
8	0.300E-02
9	0.600E-02
10	0.900E-02

Figure 5.20 Streamlines at Re=80 for a Newtonian fluid.

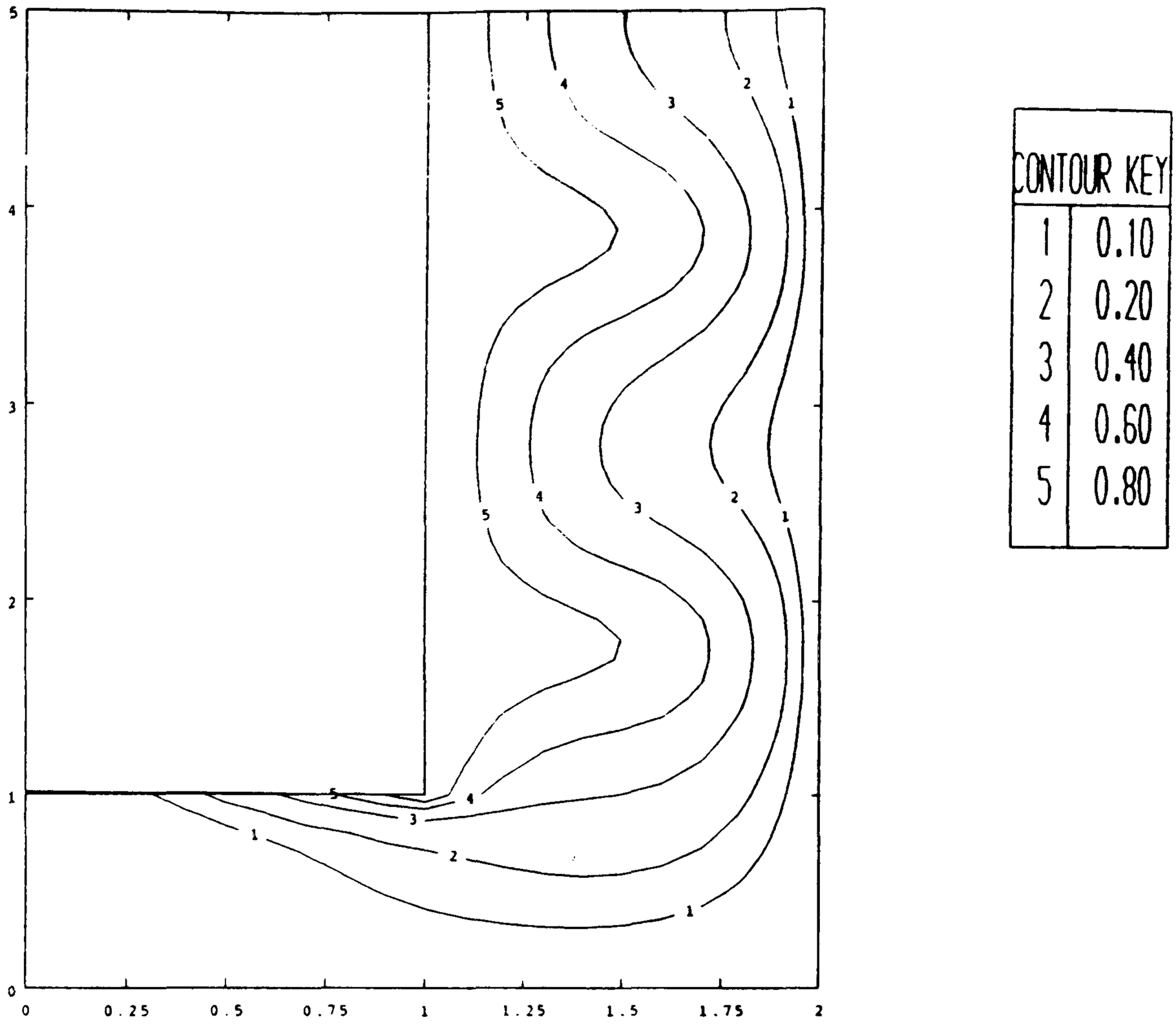


Figure 5.21 Plot of rV contours at Re=80 for a Newtonian fluid produced by Bodalia's program.

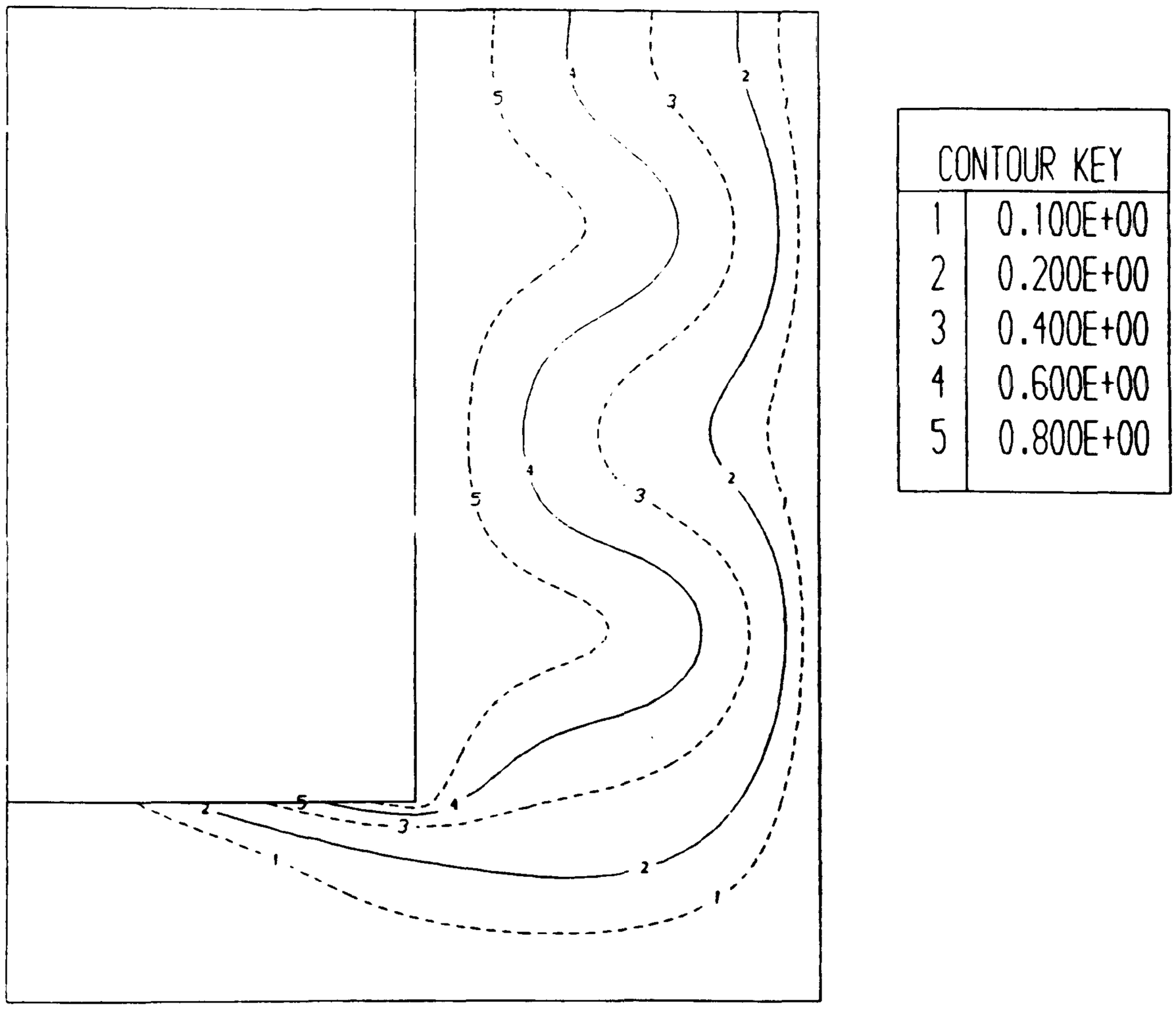
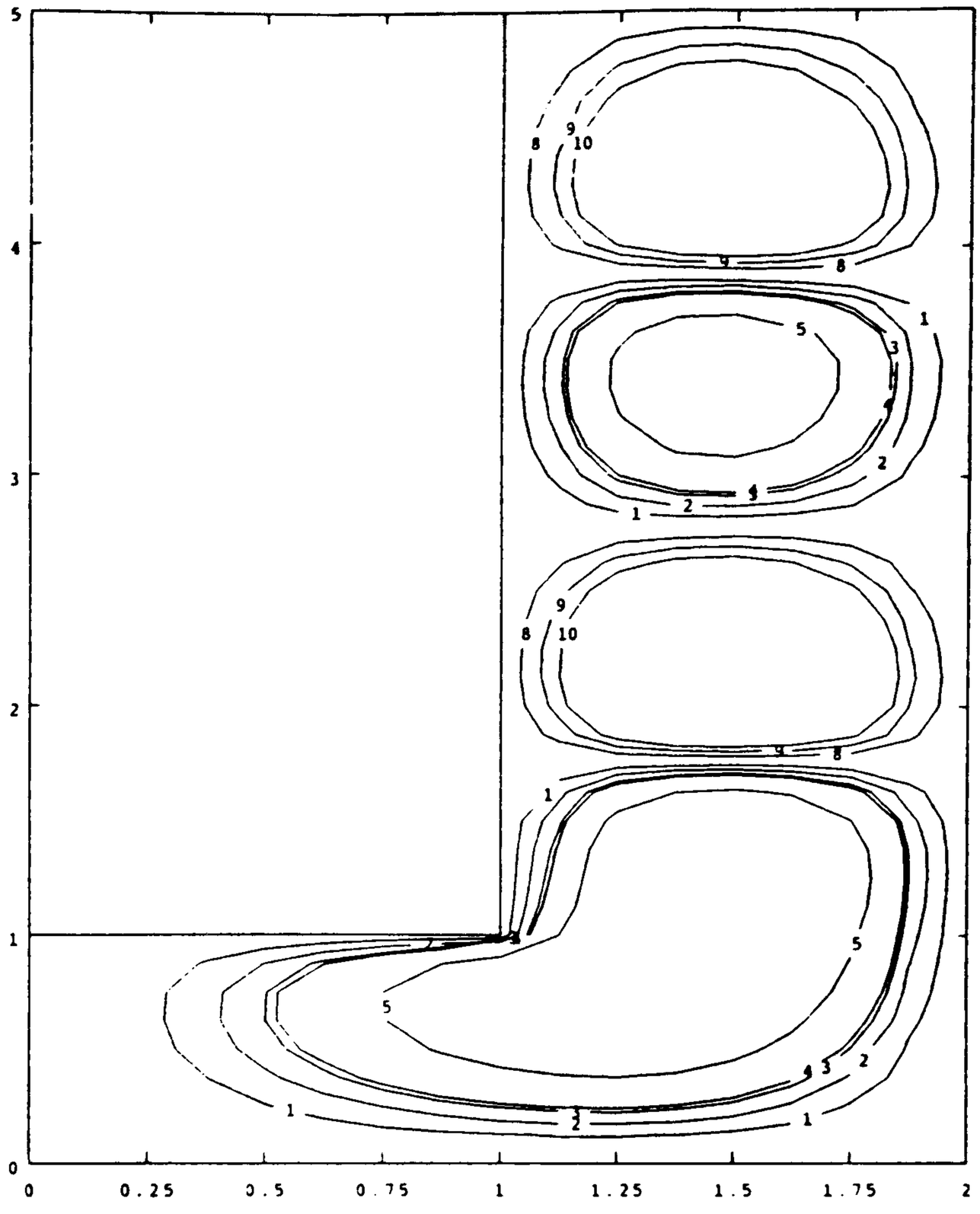
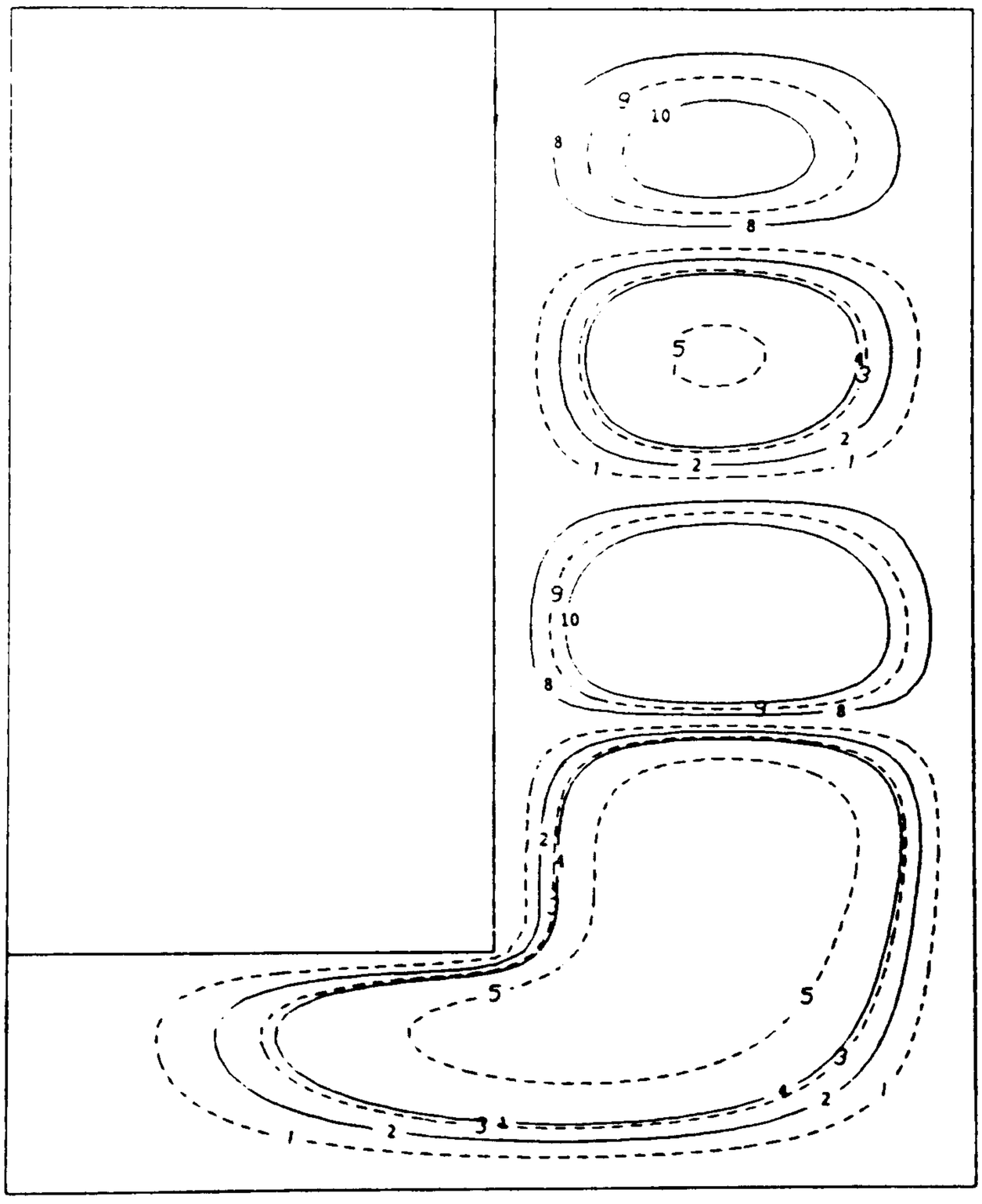


Figure 5.22 Plot of rV contours at Re=80 for a Newtonian fluid.



CONTOUR KEY	
1	-0.003
2	-0.006
3	-0.009
4	-0.010
5	-0.020
6	-0.070
7	0.070
8	0.003
9	0.006
10	0.009

Figure 5.23 Streamlines at Re=40 for a variable viscosity fluid produced by Bodalia's program.



CONTOUR KEY	
1	-0.300E-02
2	-0.600E-02
3	-0.900E-02
4	-0.100E-01
5	-0.200E-01
6	-0.700E-01
7	0.700E-01
8	0.300E-02
9	0.600E-02
10	0.900E-02

Figure 5.24 Streamlines at Re=40 for a variable viscosity fluid.

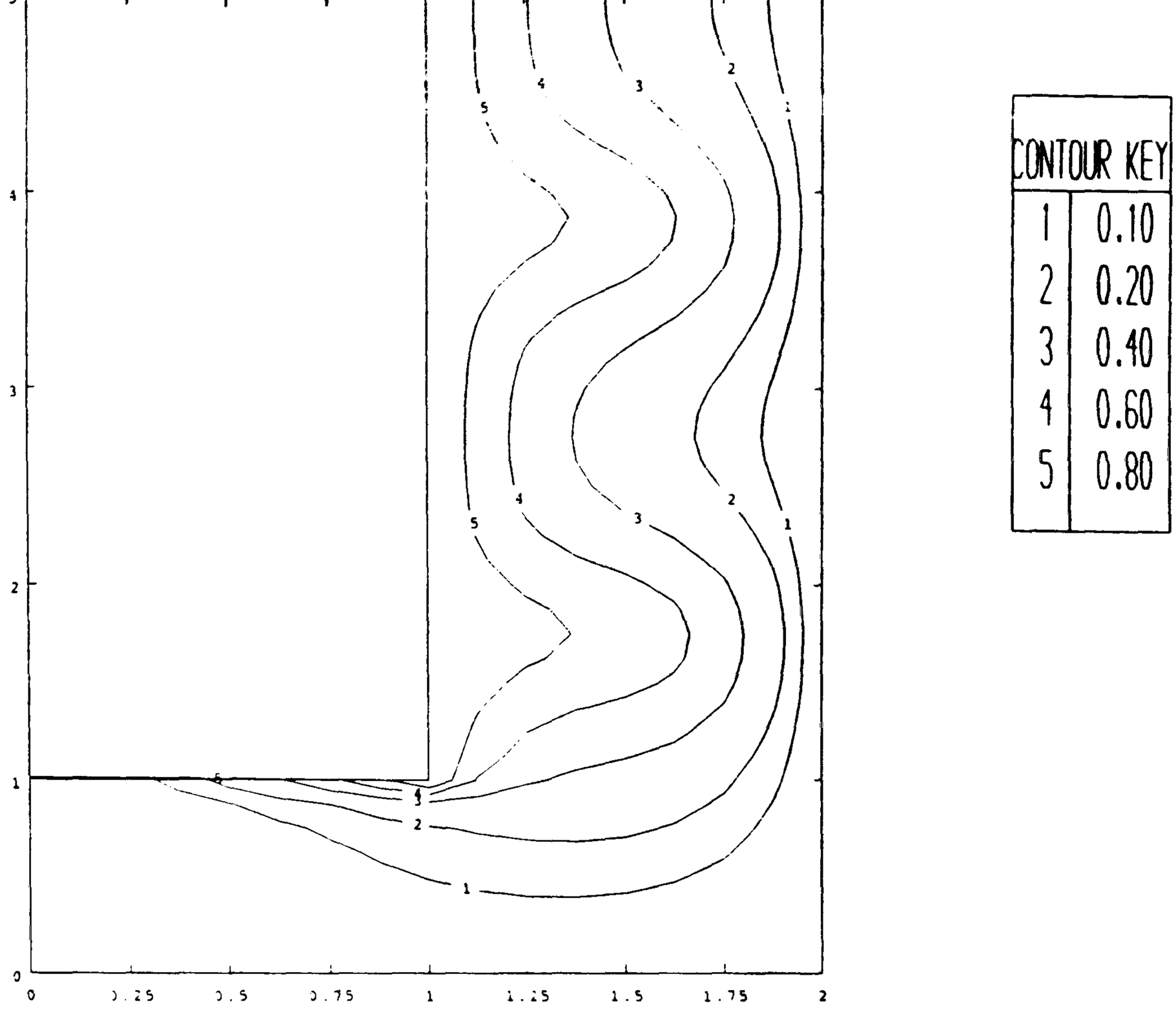


Figure 5.25 Plot of rV contours at Re=40 for a variable viscosity fluid produced by Bodalia's program.

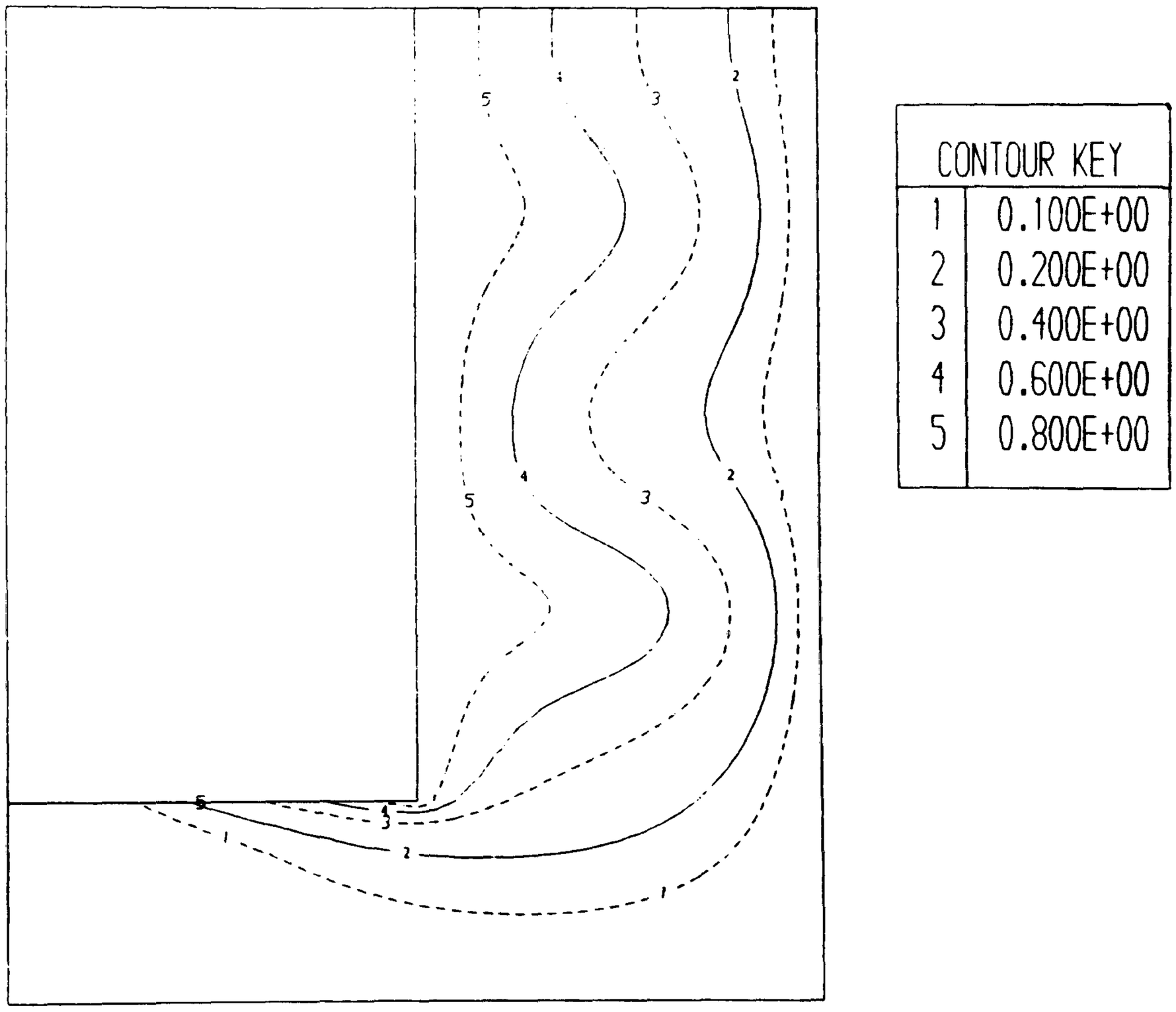


Figure 5.26 Plot of rV contours at Re=40 for a variable viscosity fluid.

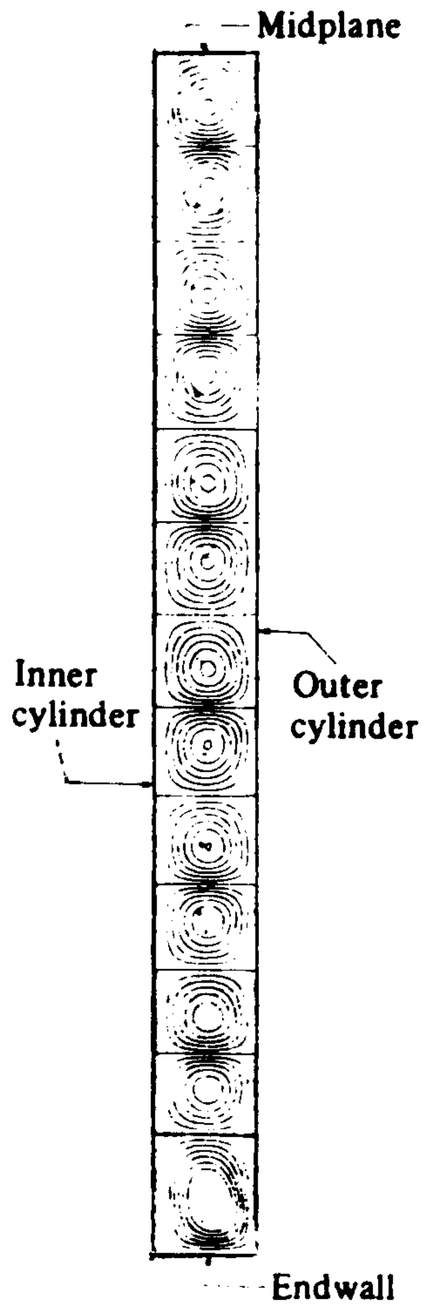


Figure 5.27 Steady state vortex pattern for Re=62.06 by Neitzel.

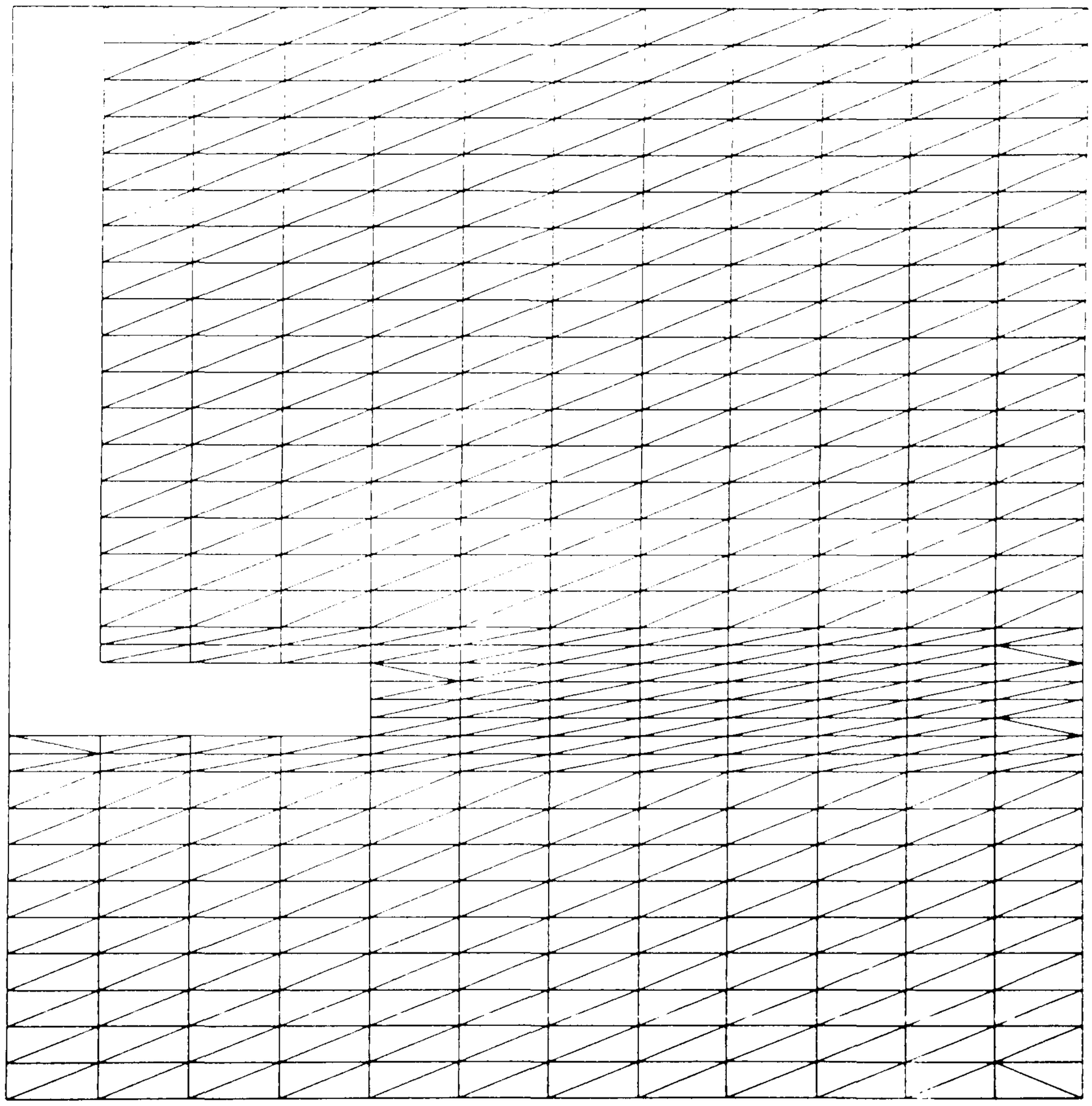


Figure 5.29 Finite element mesh with extra refinement in the vicinity of the disc.

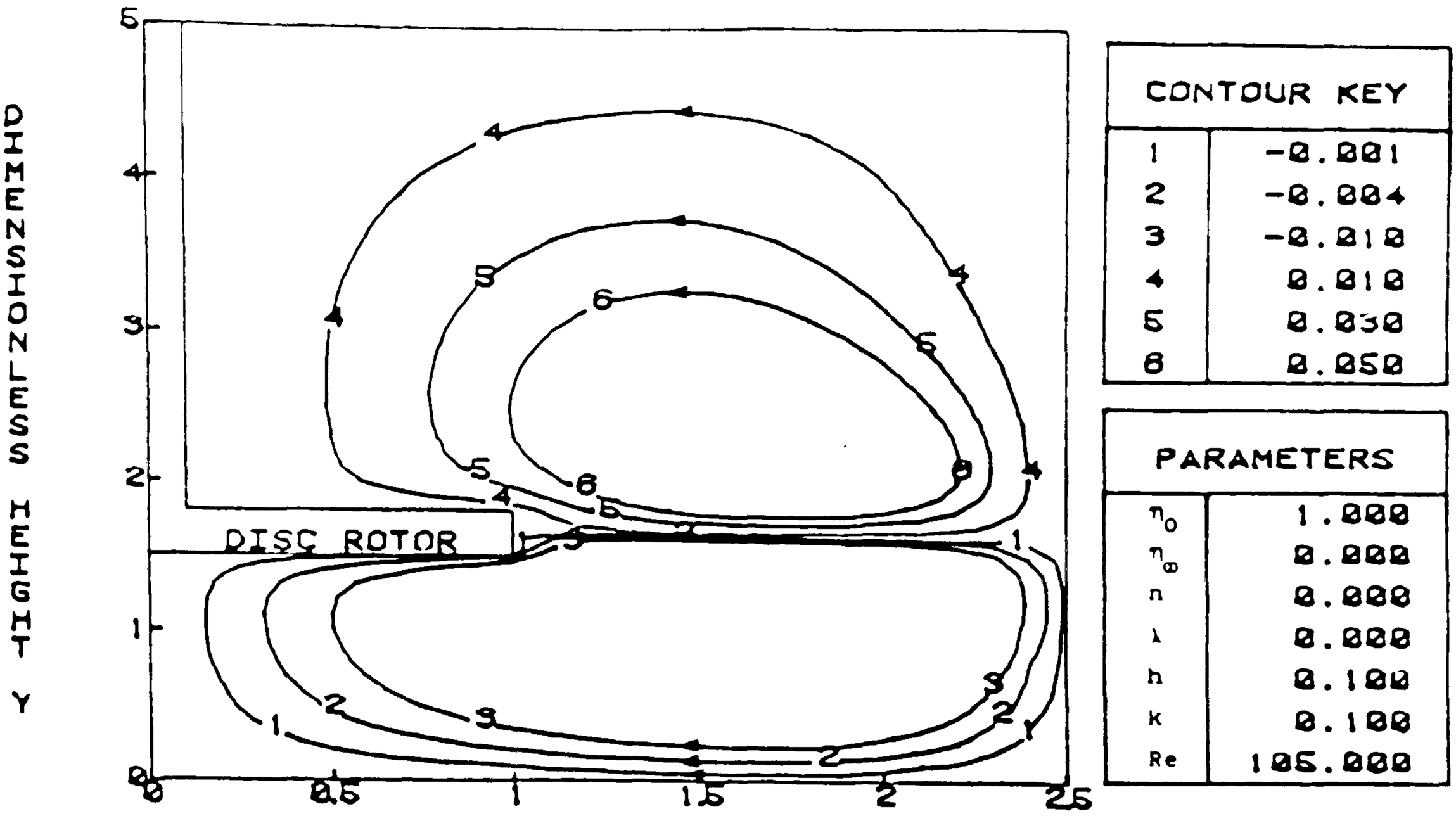


Figure 5.30 Streamlines at Re=105 for a Newtonian fluid by Bodalia.

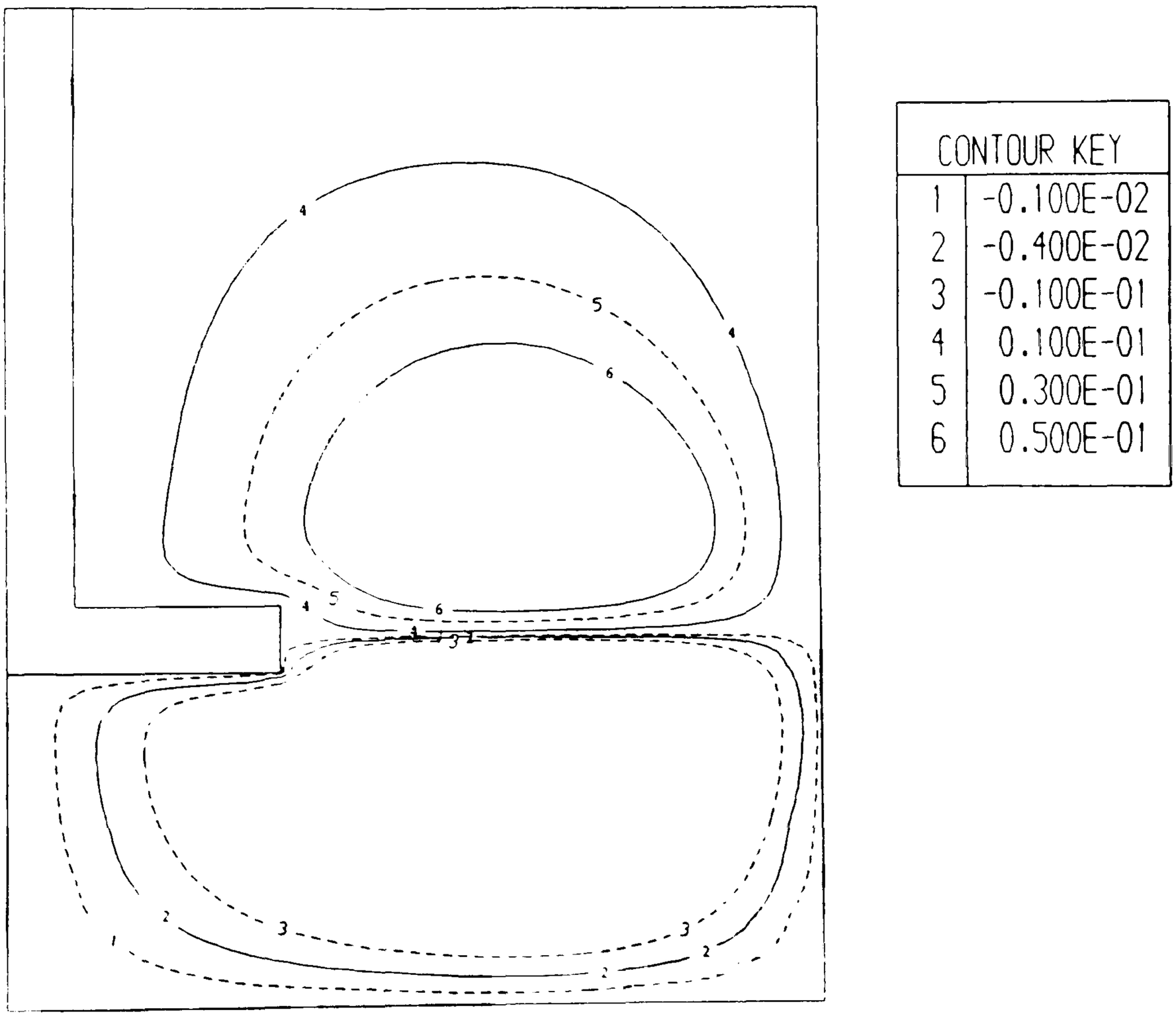


Figure 5.31 Streamlines at Re=100 for a Newtonian fluid.

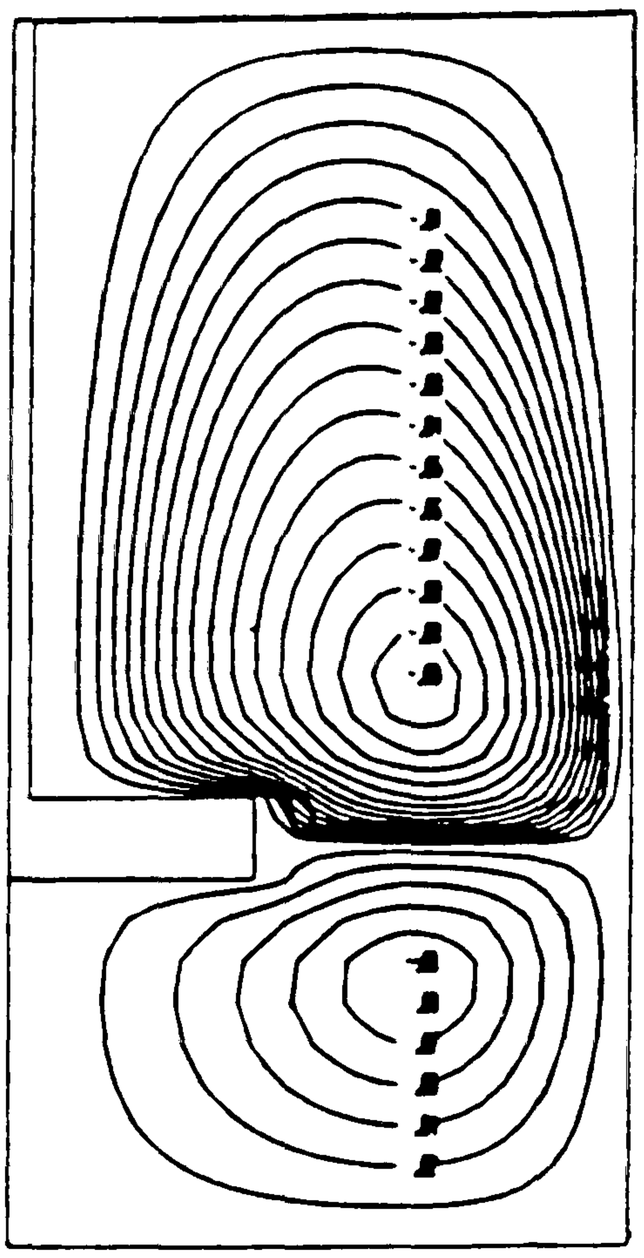


Figure 5.32 Streamlines at Re=67 for a Newtonian fluid by Spragg et al.

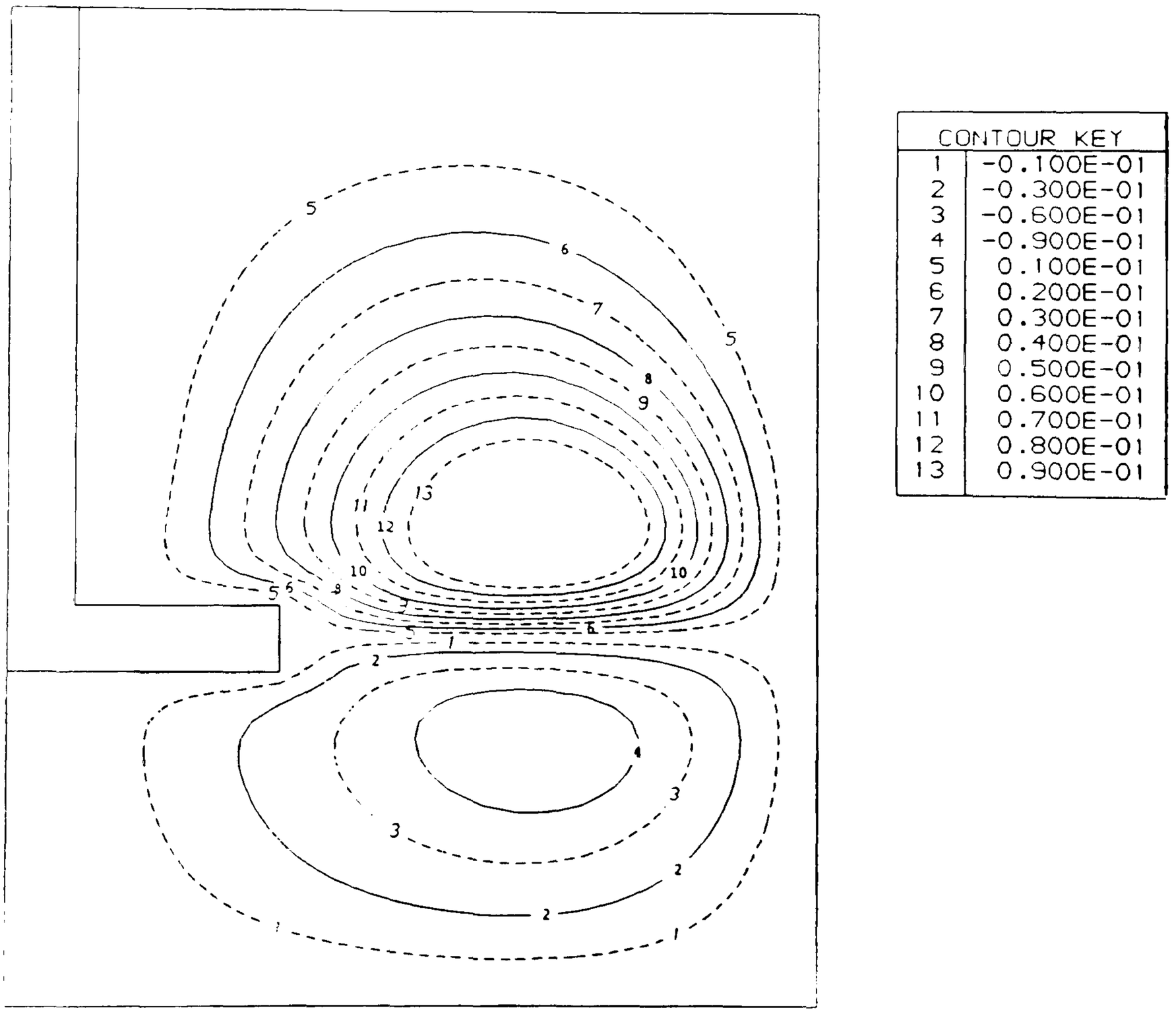


Figure 5.33 Streamlines at Re=100 for a Newtonian fluid.

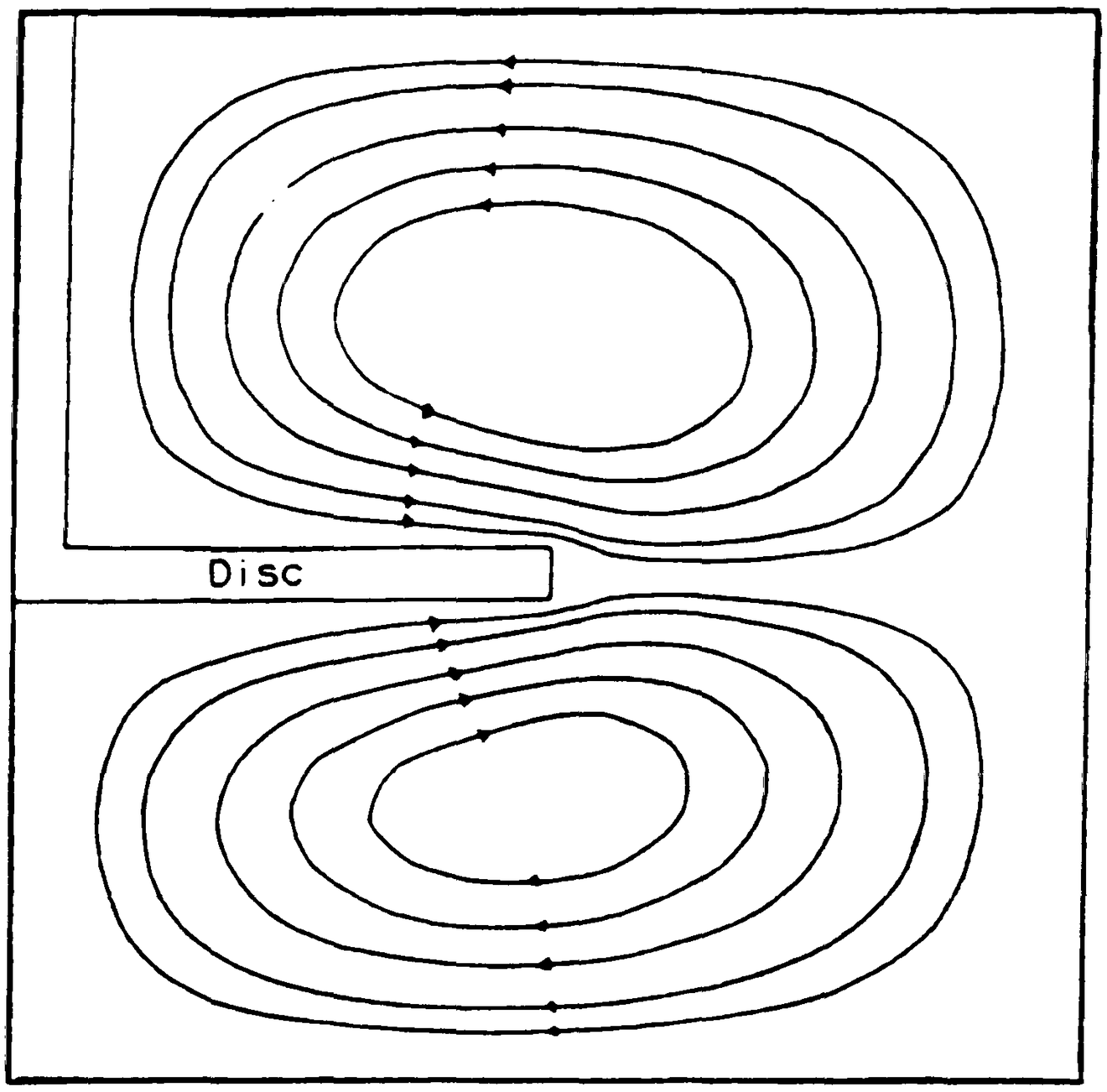
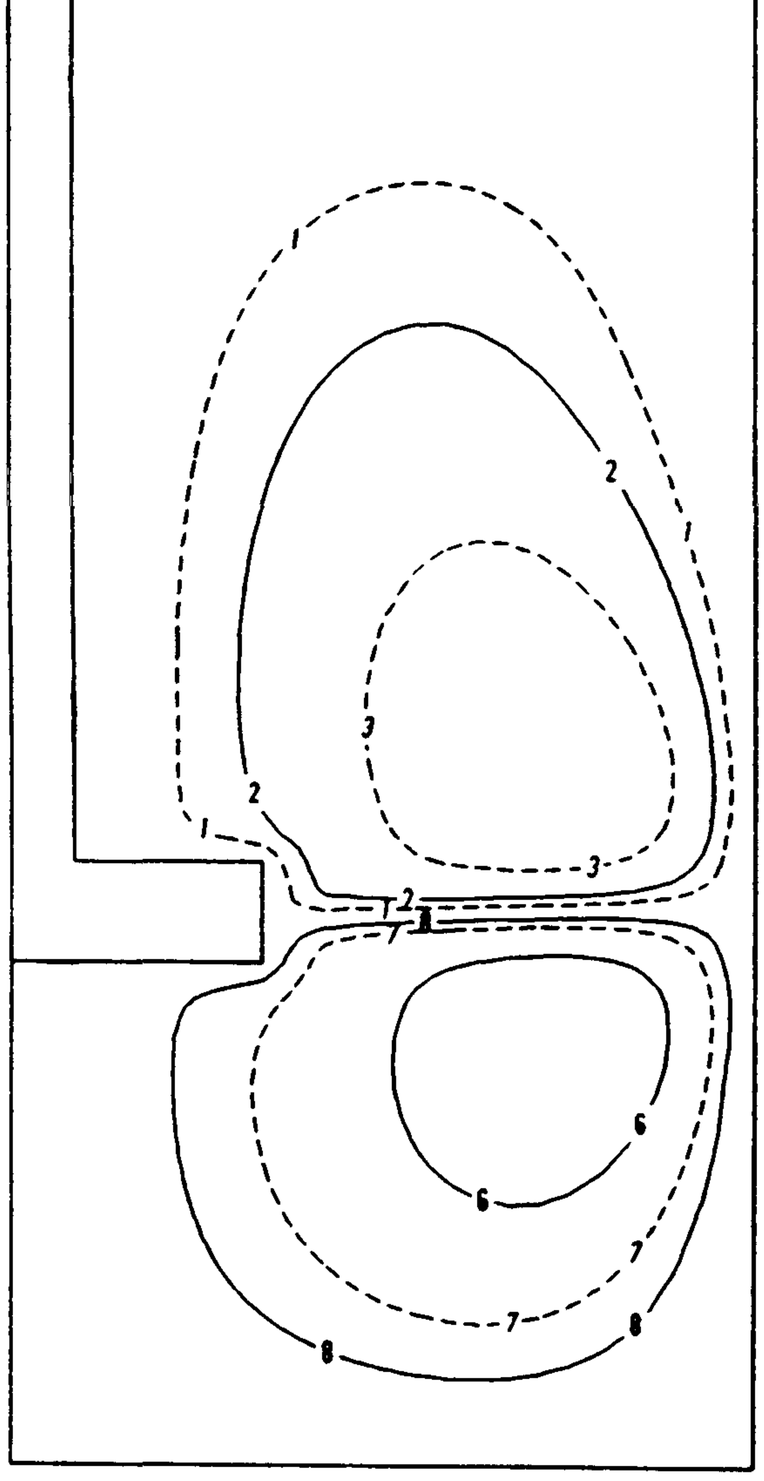
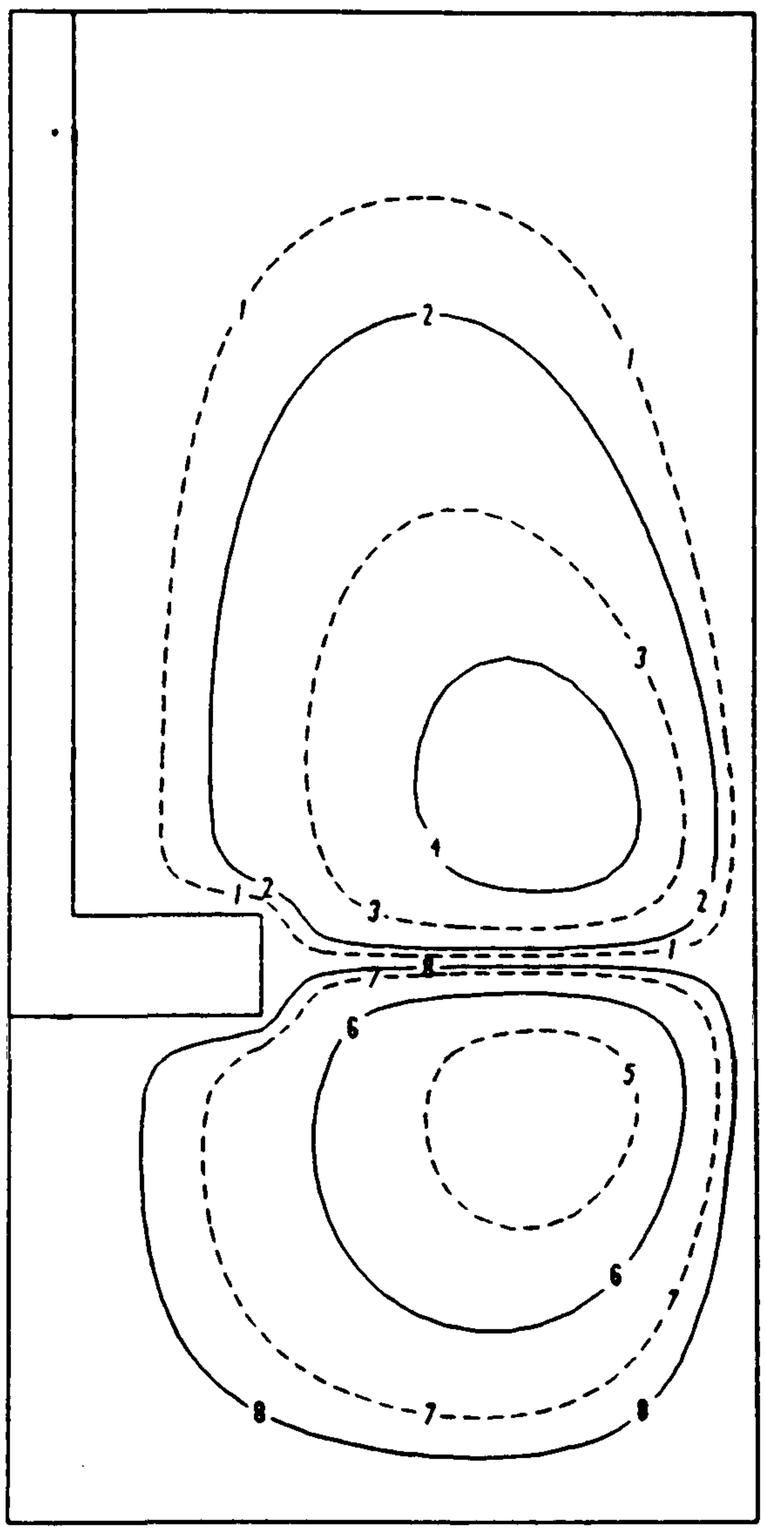


Figure 5.34 Streamlines from the work of Griffiths et al for a disc stirrer and a Newtonian fluid.



CONTOUR KEY	
1	0.900E-02
2	0.200E-01
3	0.500E-01
4	0.900E-01
5	-0.900E-01
6	-0.500E-01
7	-0.200E-01
8	-0.900E-02

Figure 5.35 Streamlines at Re=200 for a variable viscosity fluid.



CONTOUR KEY	
1	0.900E-02
2	0.200E-01
3	0.500E-01
4	0.900E-01
5	-0.900E-01
6	-0.500E-01
7	-0.200E-01
8	-0.900E-02

Figure 5.36 Streamlines at Re=200 for a Newtonian fluid.

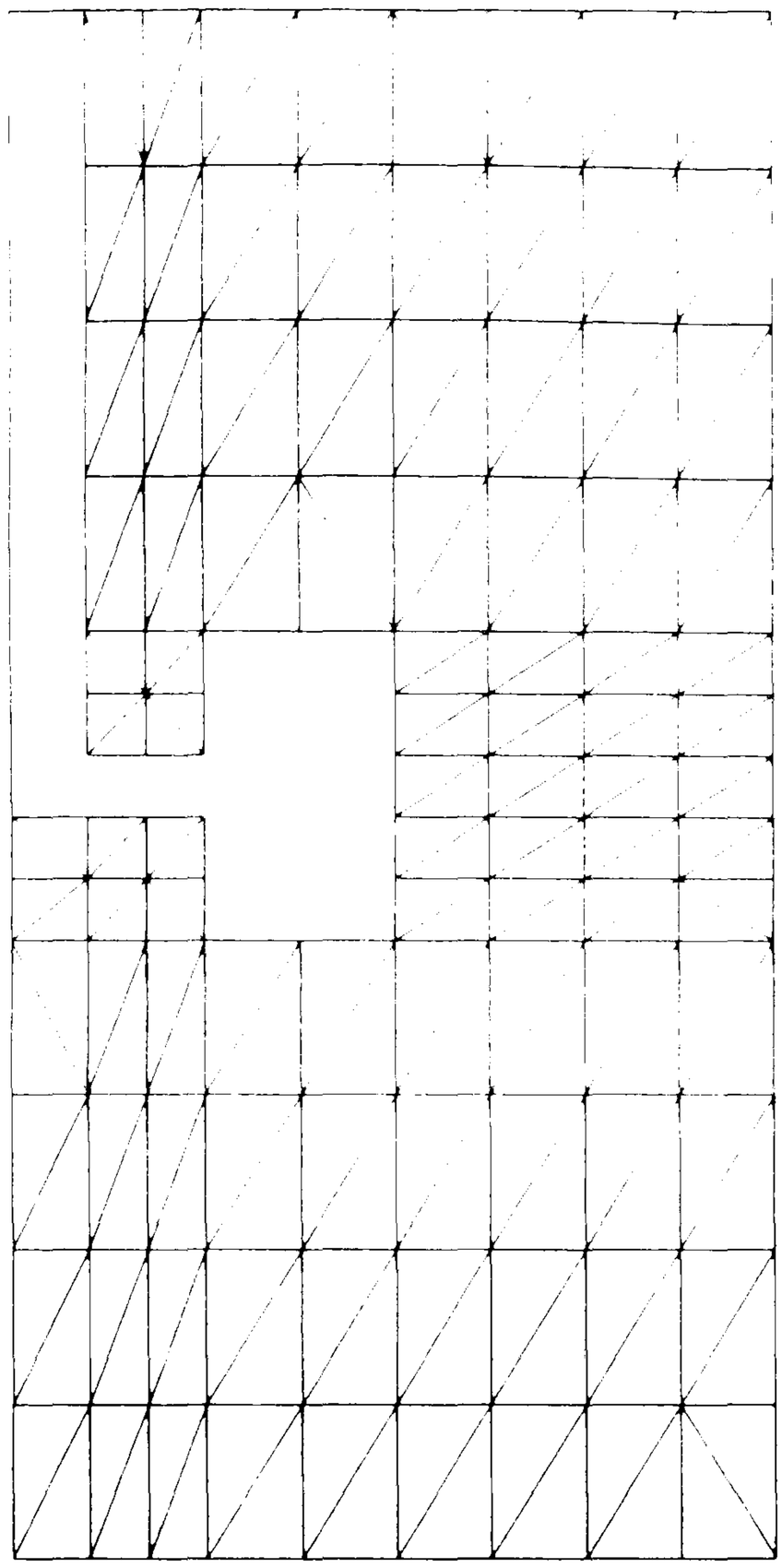
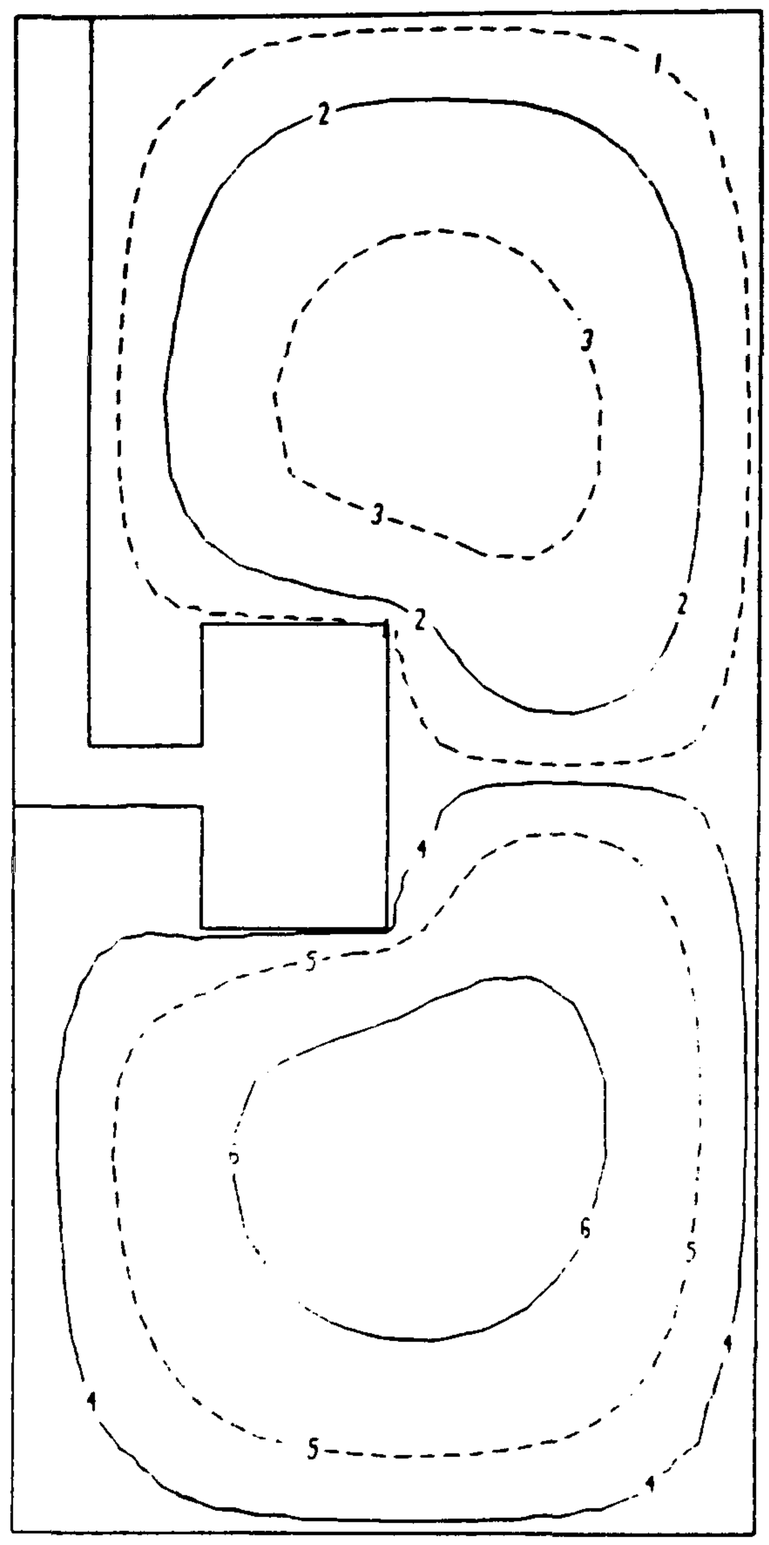
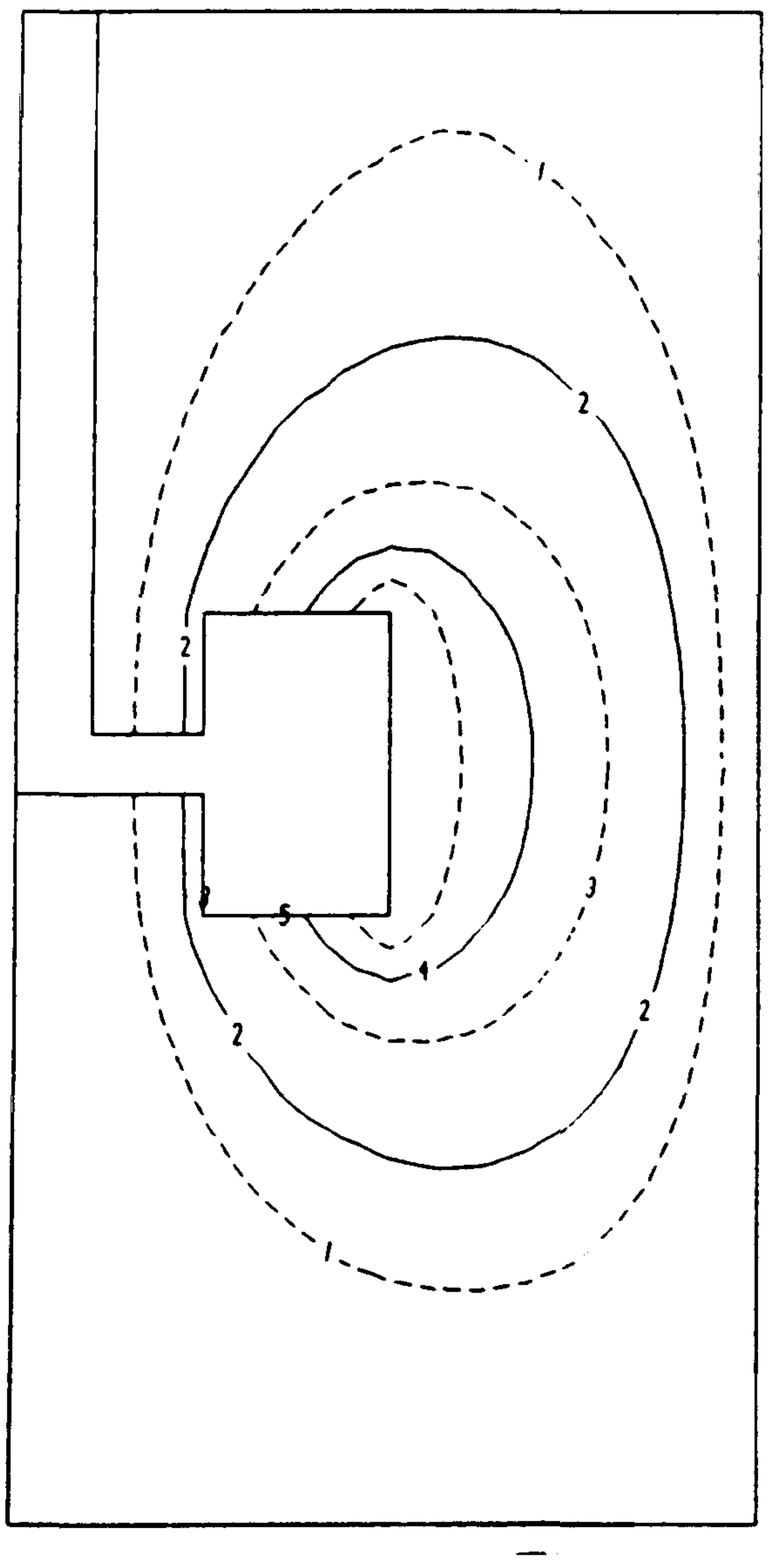


Figure 5.38 Finite element mesh with extra refinement in the vicinity of the paddle.



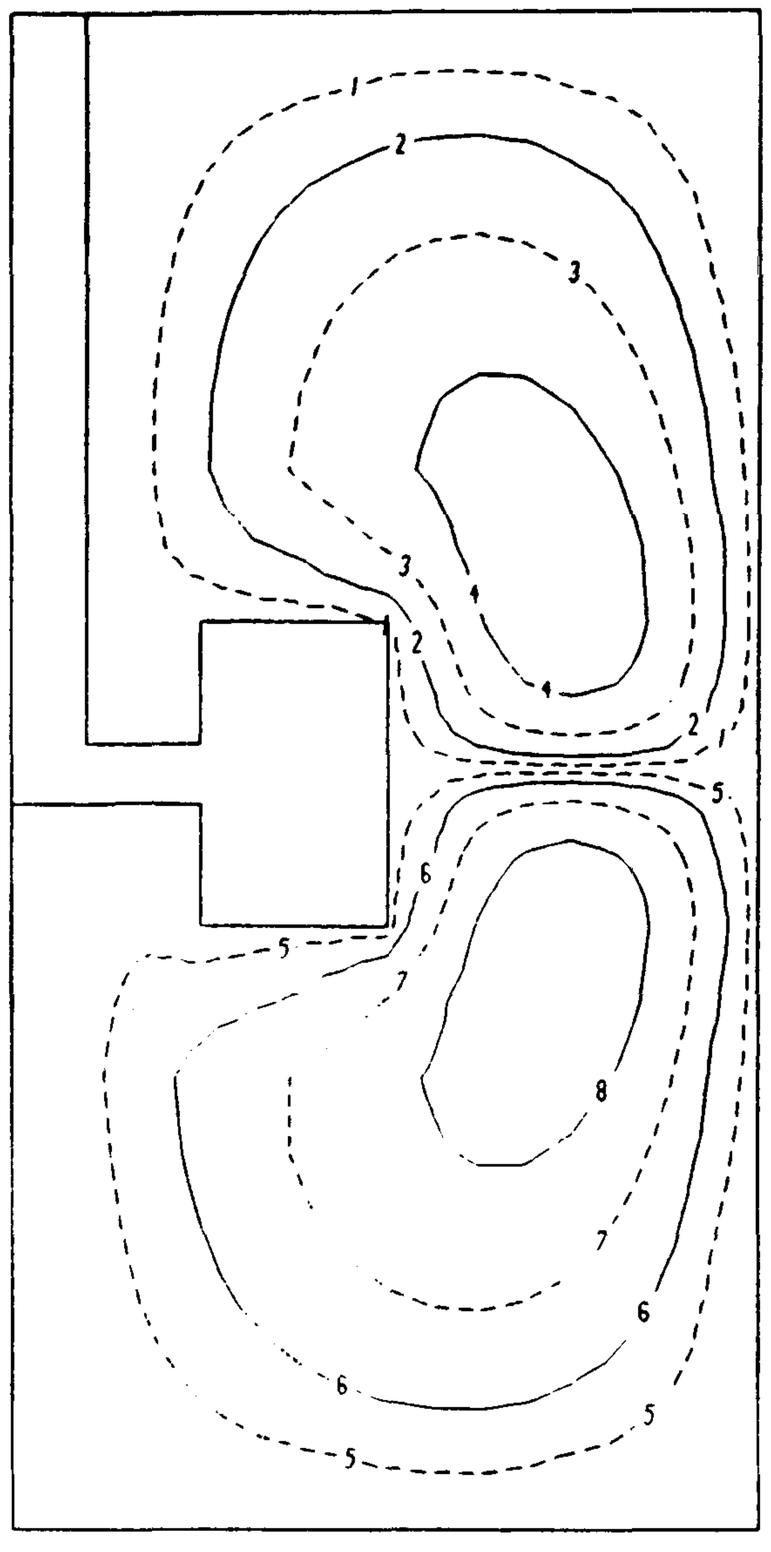
CONTOUR KEY	
1	0.300E-04
2	0.200E-03
3	0.900E-03
4	-0.300E-04
5	-0.200E-03
6	-0.900E-03

Figure 5.39 Streamlines at Re=1 for a Newtonian fluid.



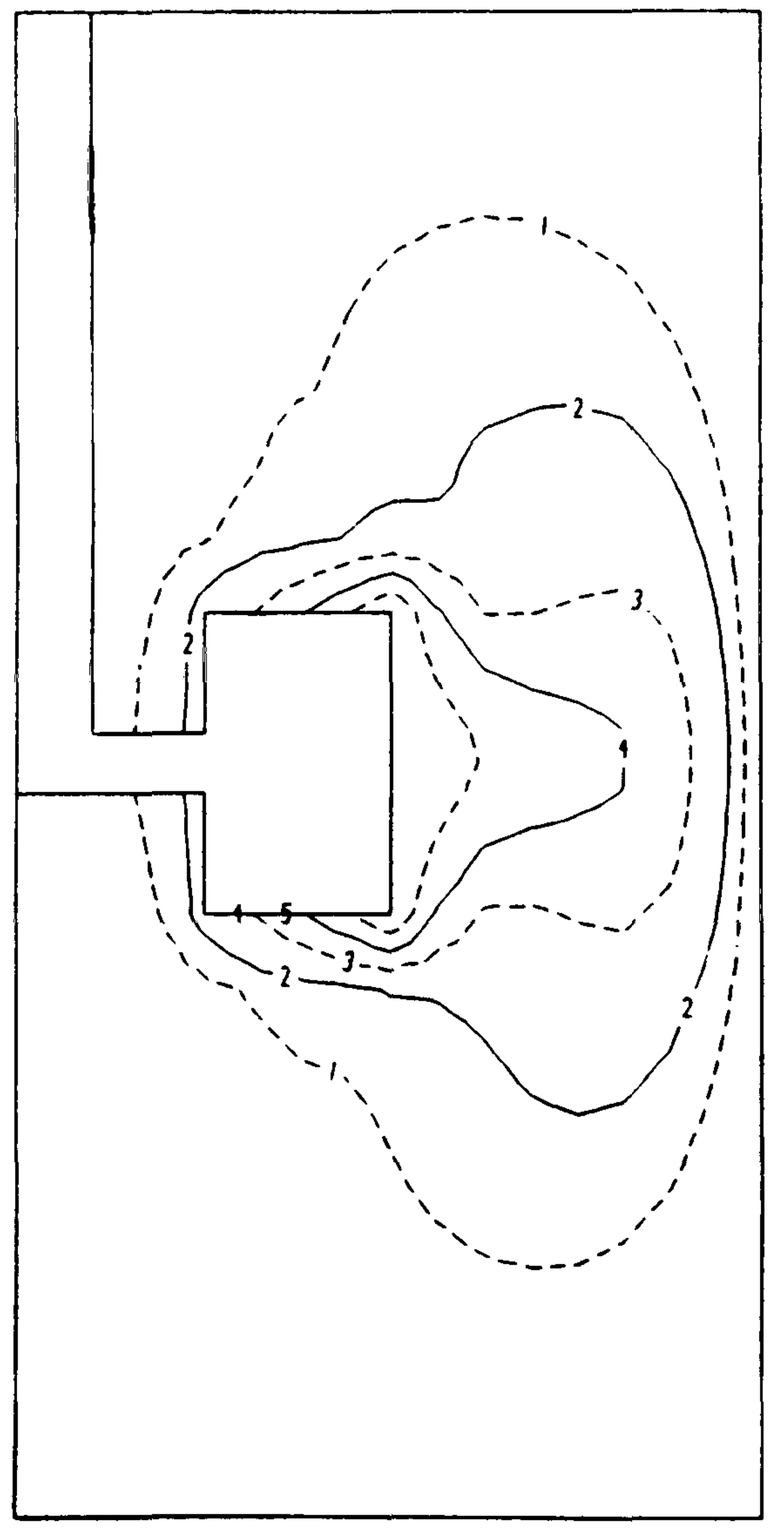
CONTOUR KEY	
1	0.100E+00
2	0.200E+00
3	0.400E+00
4	0.600E+00
5	0.800E+00

Figure 5.40 Plot of rV contours at Re=1 for a Newtonian fluid.



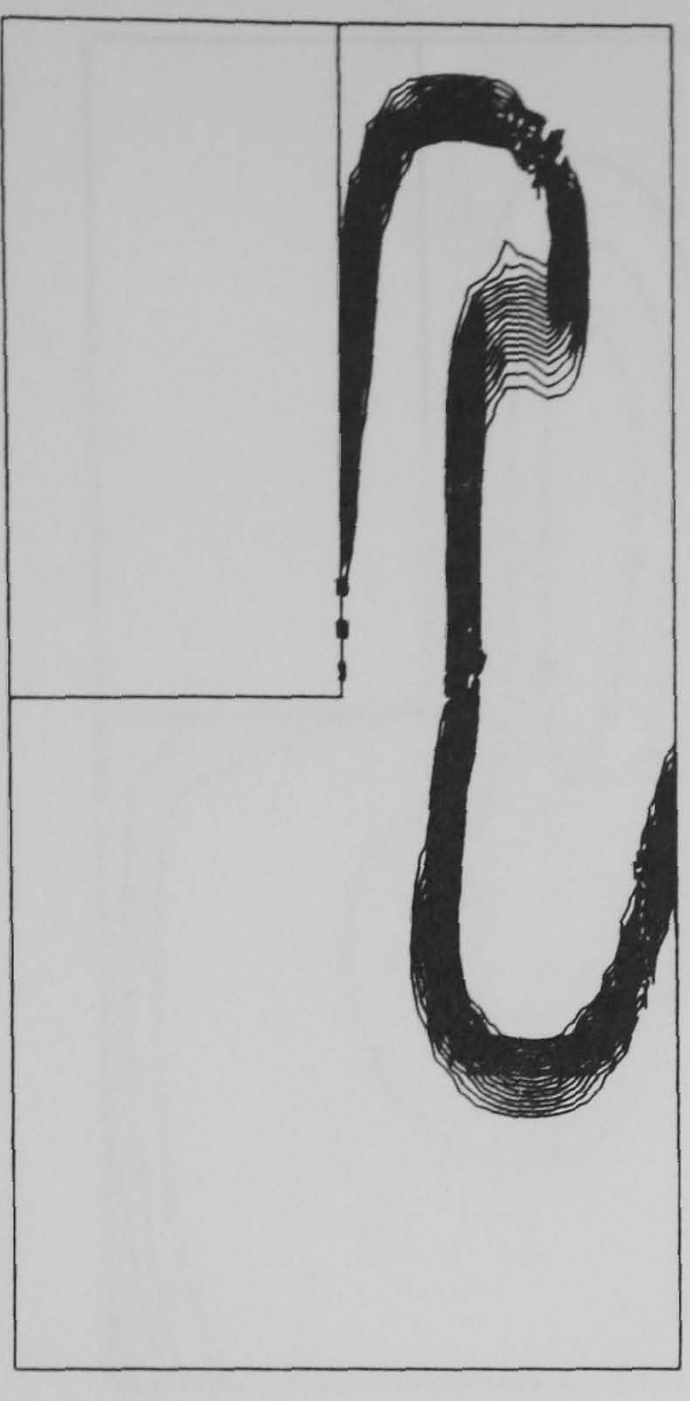
CONTOUR KEY	
1	0.250E-02
2	0.800E-02
3	0.200E-01
4	0.400E-01
5	-0.250E-02
6	-0.800E-02
7	-0.200E-01
8	-0.400E-01

Figure 5.41 Streamlines at Re=100 for a Newtonian fluid.

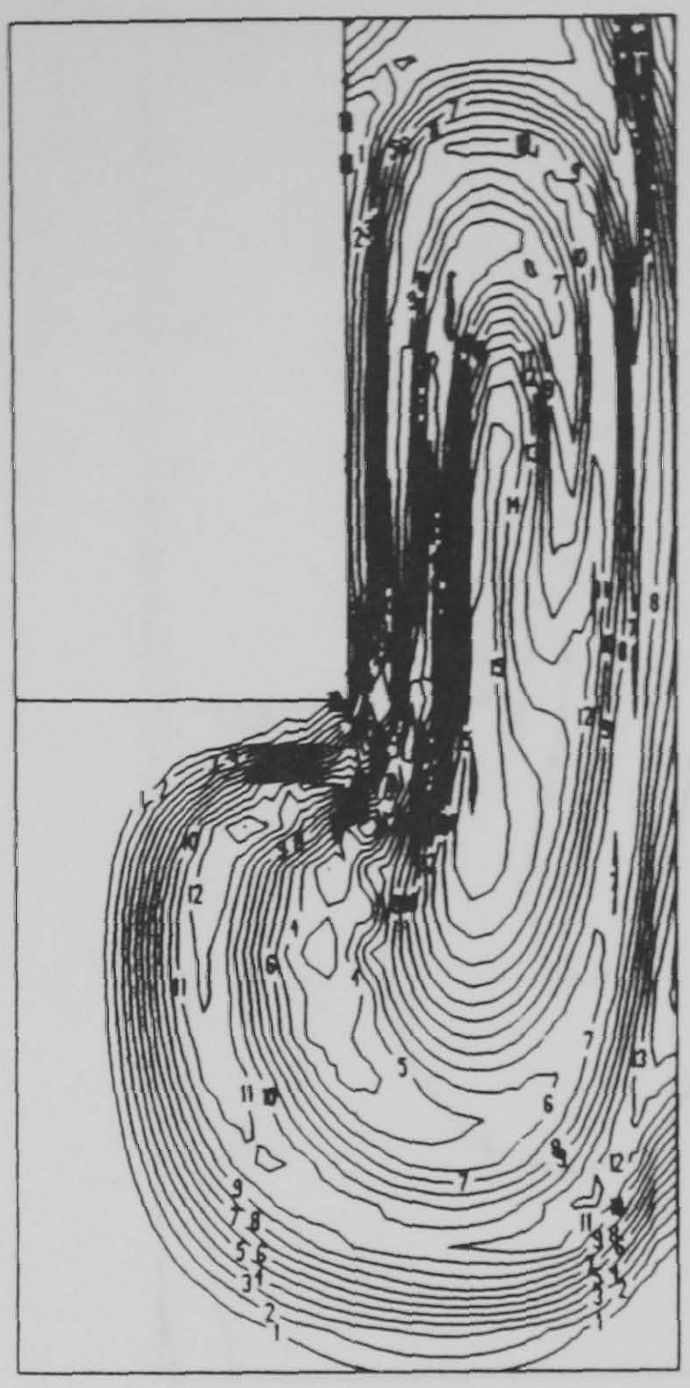


CONTOUR KEY	
1	0.100E+00
2	0.200E+00
3	0.400E+00
4	0.600E+00
5	0.800E+00

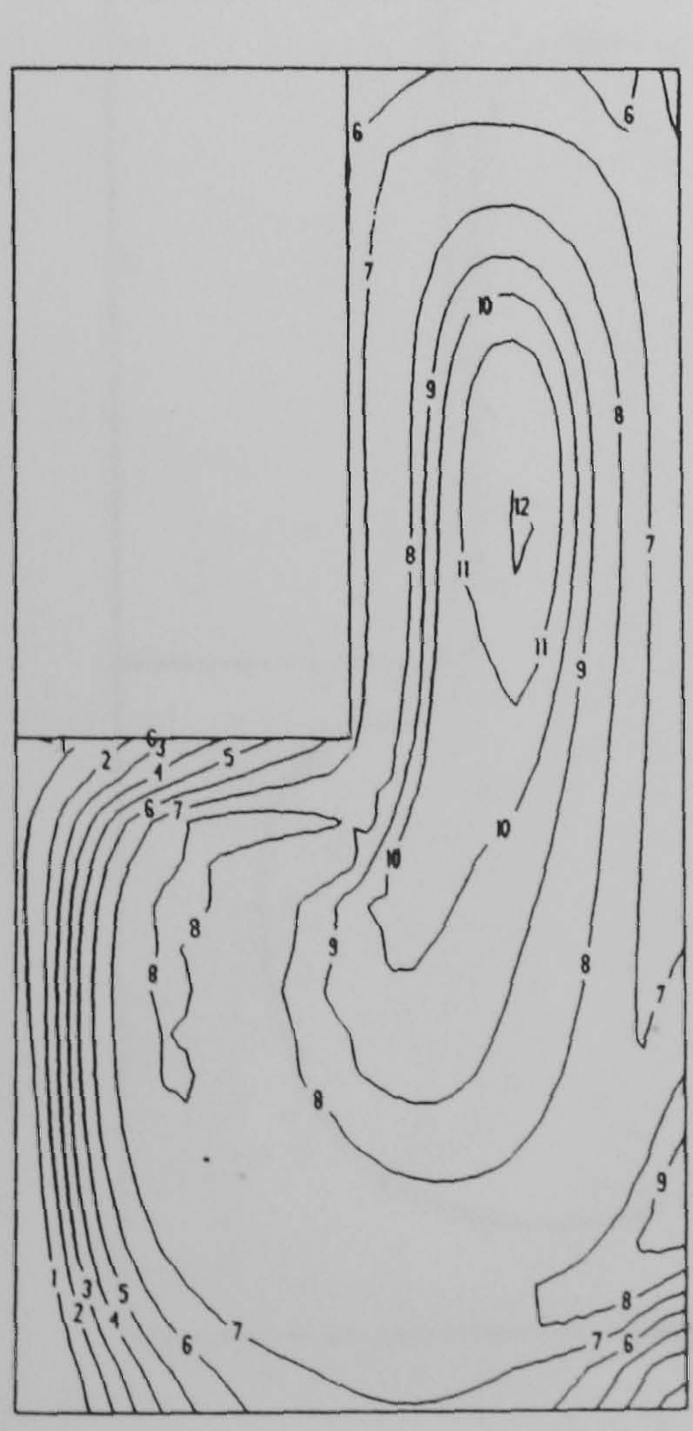
Figure 5.42 Plot of rV contours at Re=100 for a Newtonian fluid.



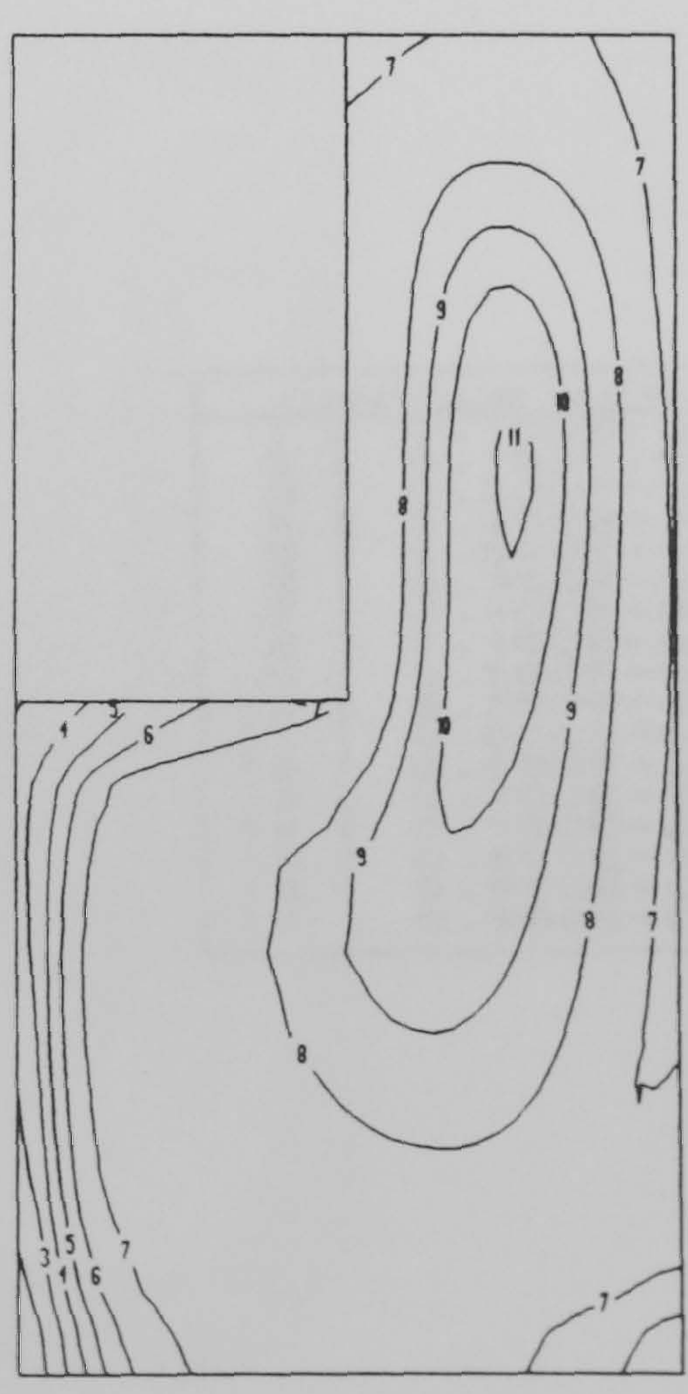
(a) after 10 secs



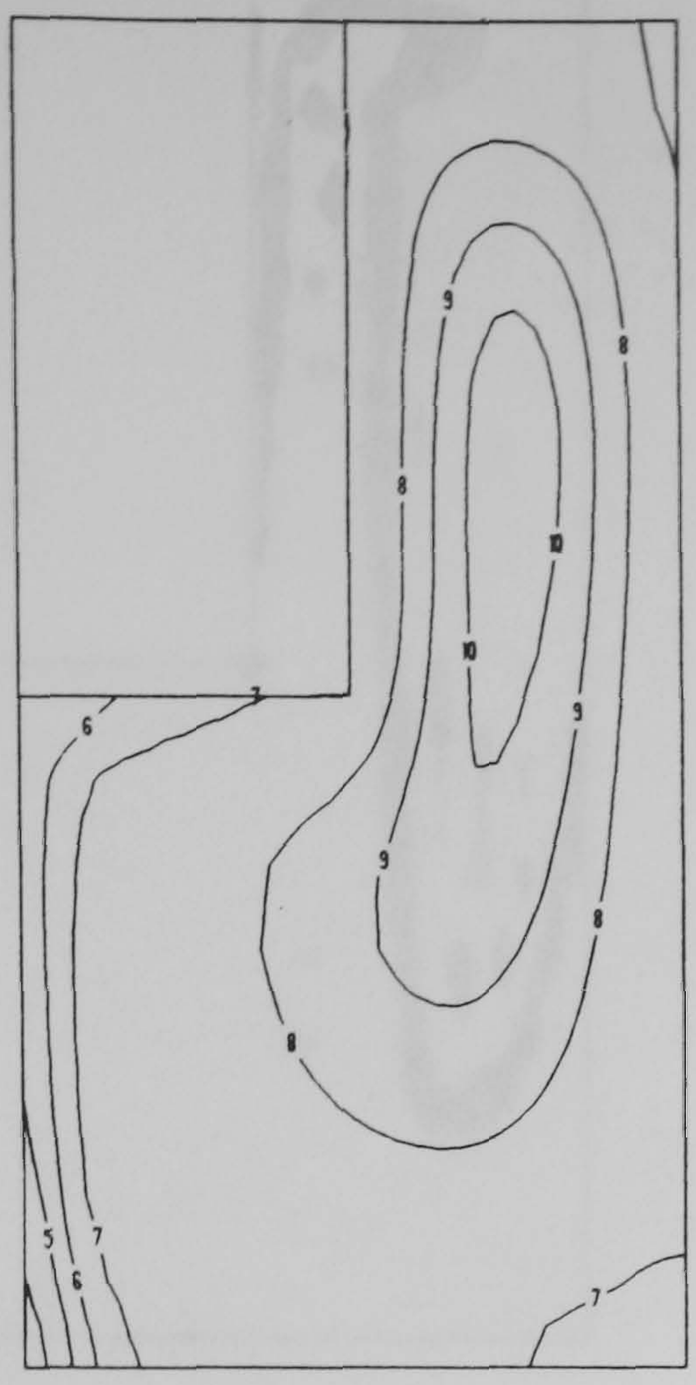
(b) after 50 secs



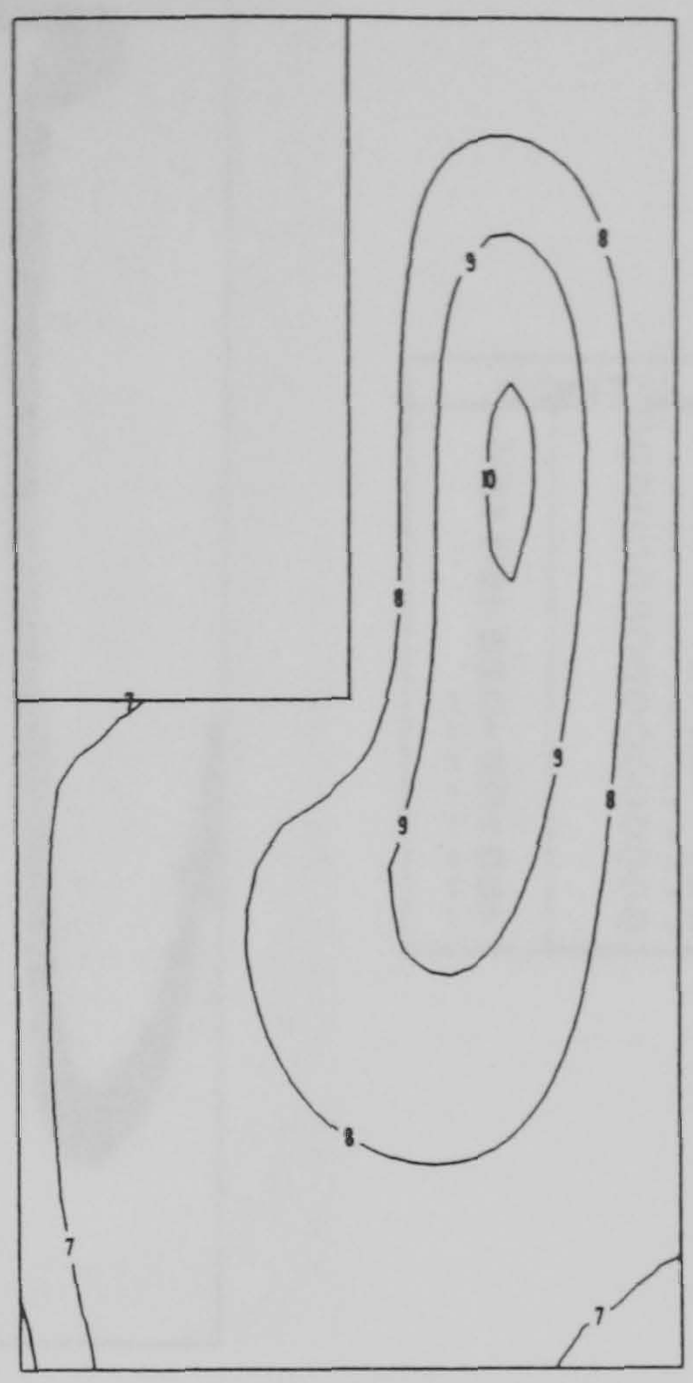
(c) after 100 secs



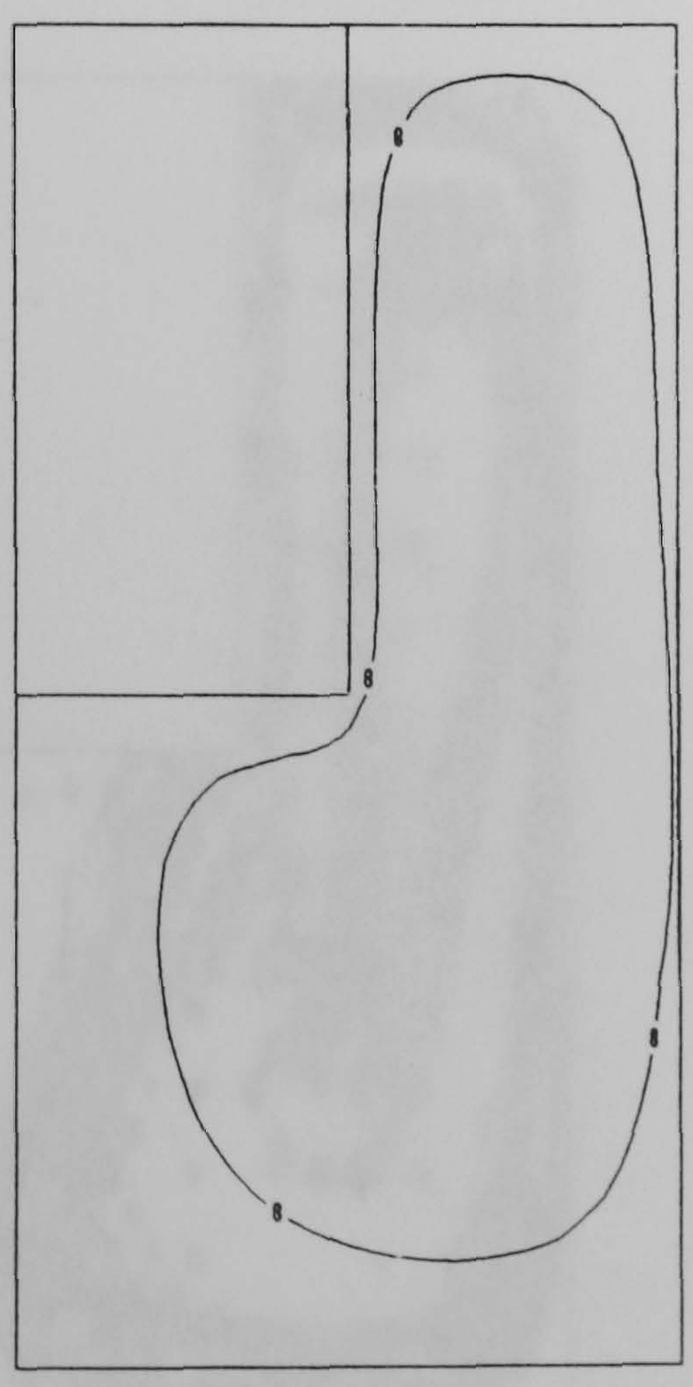
(d) after 133.4 secs (which is a 25% mix)



(e) after 160 secs (which is a 50% mix)



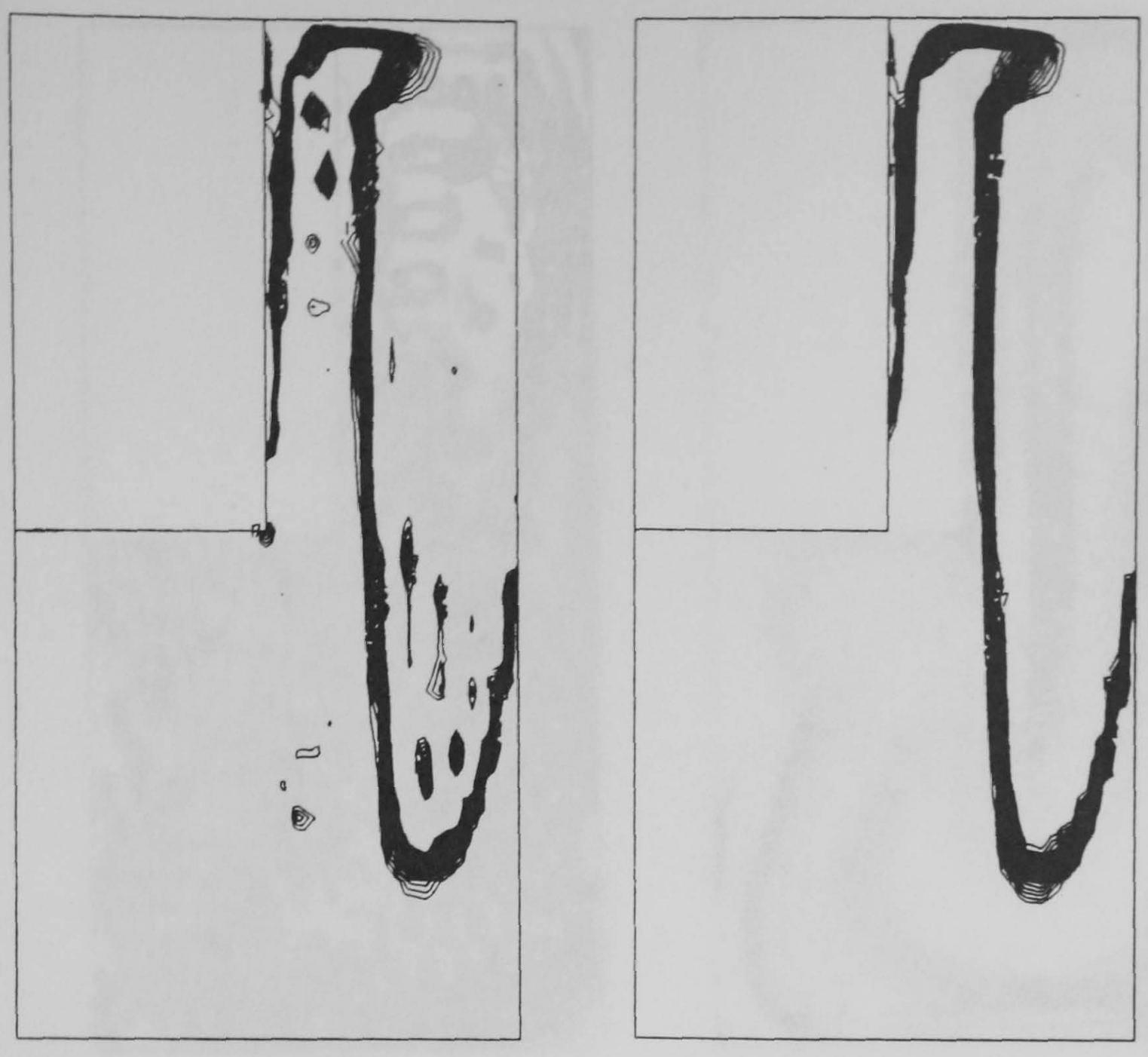
(f) after 195.4 secs (which is a 75% mix)



CONTOUR KEY	
1	0.100E+00
2	0.150E+00
3	0.200E+00
4	0.250E+00
5	0.300E+00
6	0.350E+00
7	0.400E+00
8	0.450E+00
9	0.500E+00
10	0.550E+00
11	0.600E+00
12	0.650E+00
13	0.700E+00
14	0.750E+00
15	0.800E+00
16	0.850E+00
17	0.900E+00

(g) after 432.6 secs (which is a 95% mix)

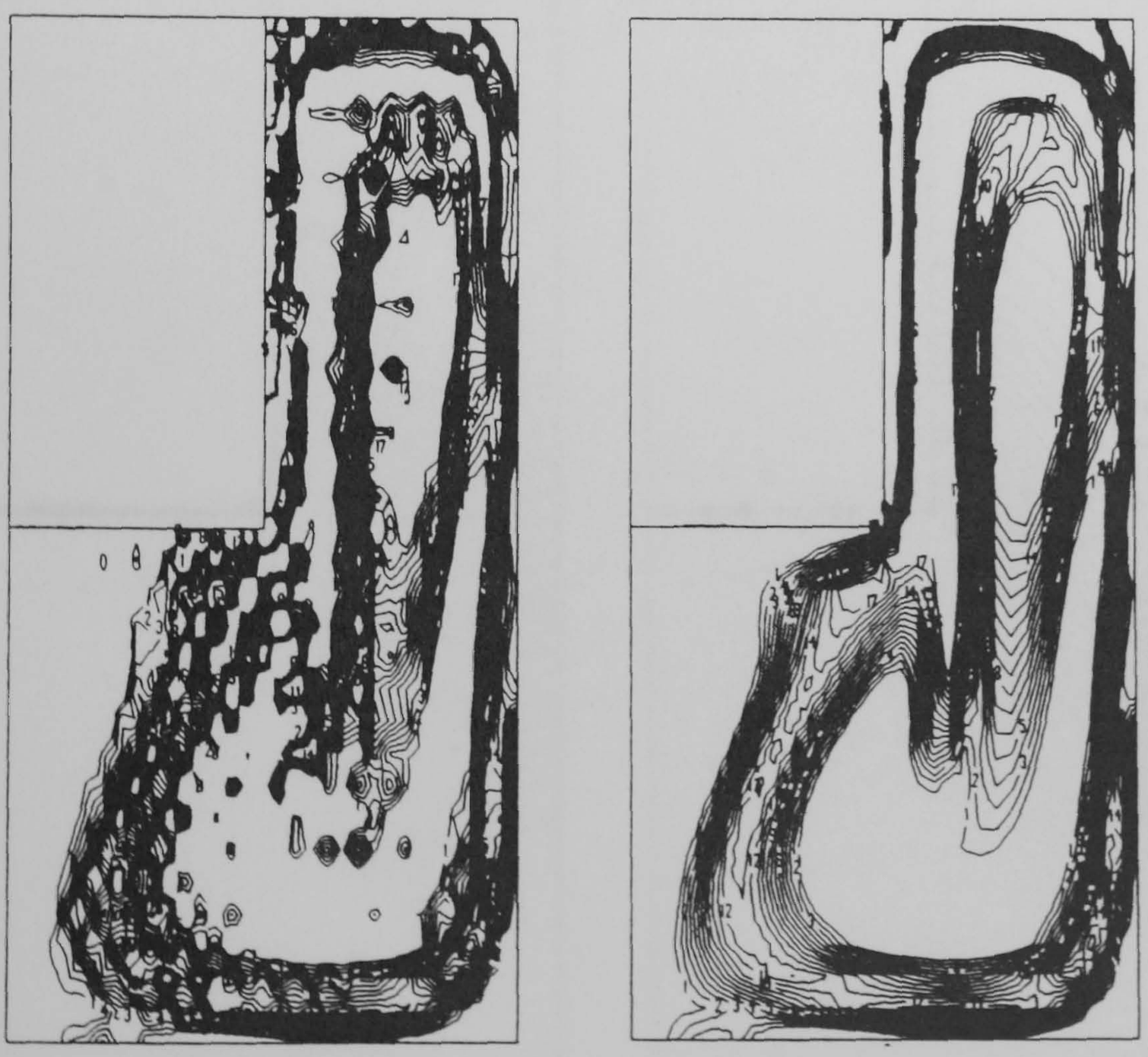
Figure 5.44 Concentration fields at Re=100 for a Newtonian fluid, produced by a non-upwinded approach.



CONTOUR KEY	
1	0.100E+00
2	0.150E+00
3	0.200E+00
4	0.250E+00
5	0.300E+00
6	0.350E+00
7	0.400E+00
8	0.450E+00
9	0.500E+00
10	0.550E+00
11	0.600E+00
12	0.650E+00
13	0.700E+00
14	0.750E+00
15	0.800E+00
16	0.850E+00
17	0.900E+00

(a) (b)

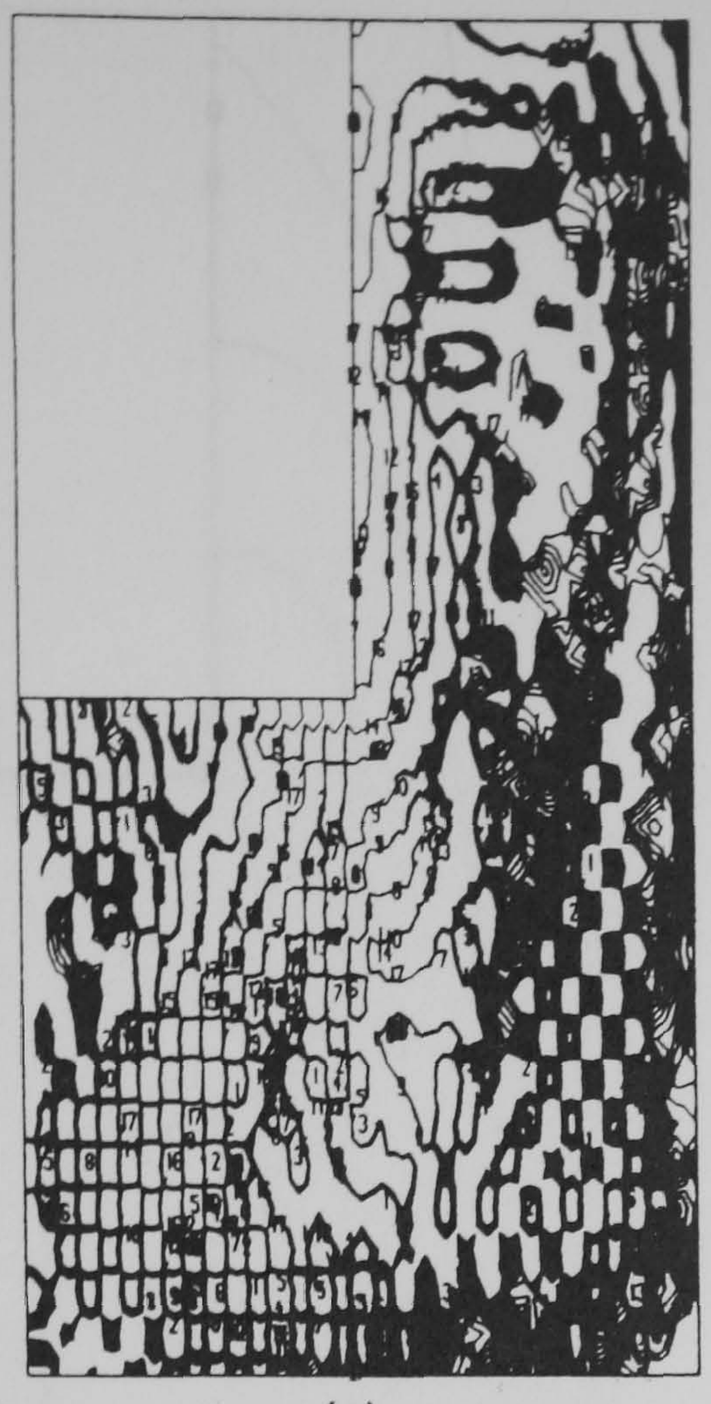
Figure 5.45 Concentration fields at Re=800 for a Newtonian fluid after 10 secs, produced by; (a) a non-upwinded approach and (b) the SU/PG approach.



CONTOUR KEY	
1	0.100E+00
2	0.150E+00
3	0.200E+00
4	0.250E+00
5	0.300E+00
6	0.350E+00
7	0.400E+00
8	0.450E+00
9	0.500E+00
10	0.550E+00
11	0.600E+00
12	0.650E+00
13	0.700E+00
14	0.750E+00
15	0.800E+00
16	0.850E+00
17	0.900E+00

(a) (b)

Figure 5.46 Concentration fields at Re=800 for a Newtonian fluid after 50 secs, produced by; (a) a non-upwinded approach and (b) the SU/PG approach.

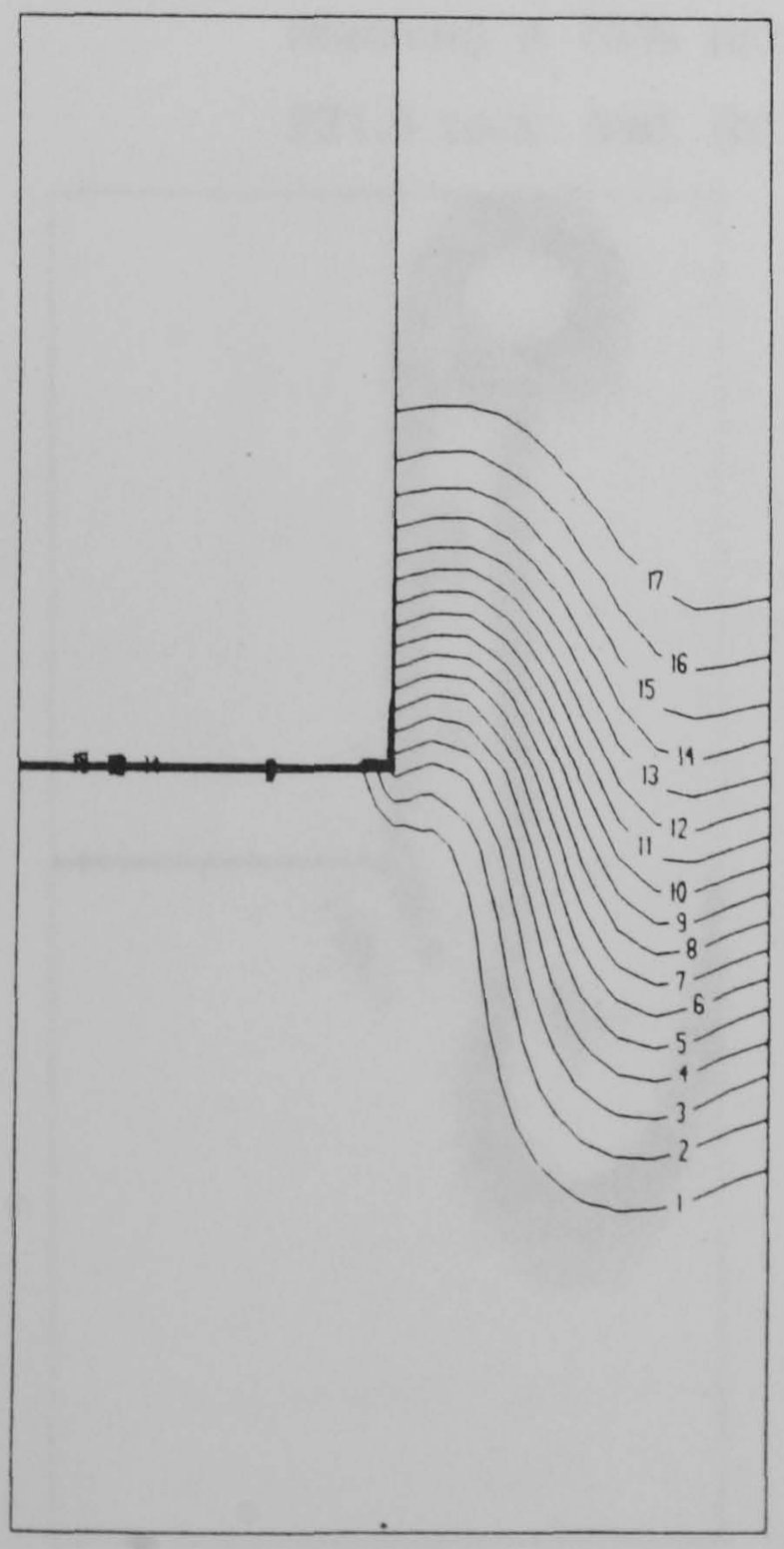


(a)

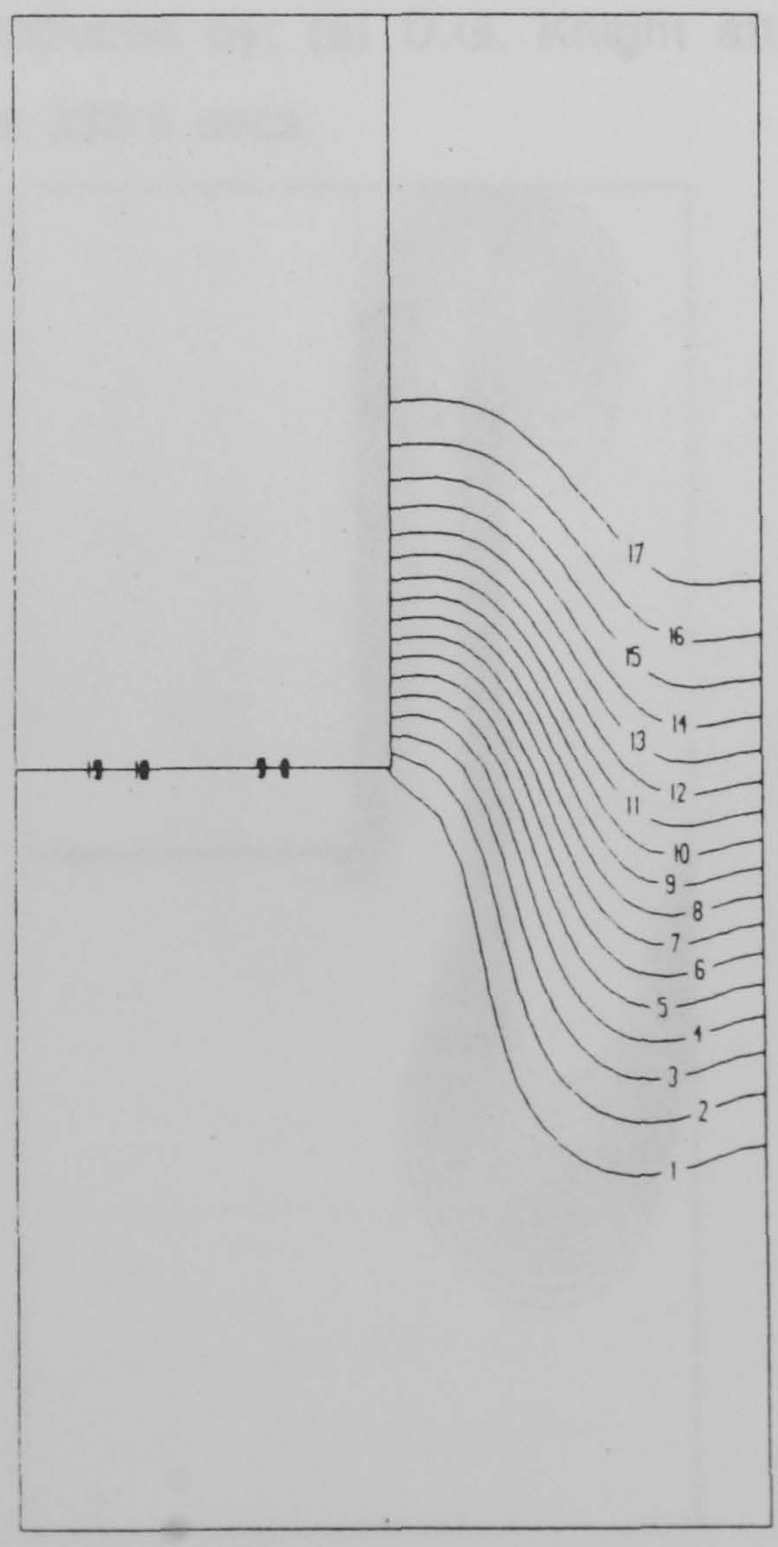
(b)

CONTOUR KEY	
1	0.100E+00
2	0.150E+00
3	0.200E+00
4	0.250E+00
5	0.300E+00
6	0.350E+00
7	0.400E+00
8	0.450E+00
9	0.500E+00
10	0.550E+00
11	0.600E+00
12	0.650E+00
13	0.700E+00
14	0.750E+00
15	0.800E+00
16	0.850E+00
17	0.900E+00

Figure 5.47 Concentration fields at Re=800 for a Newtonian fluid after 100 secs, produced by; (a) a non-upwinded approach and (b) the SU/PG approach.



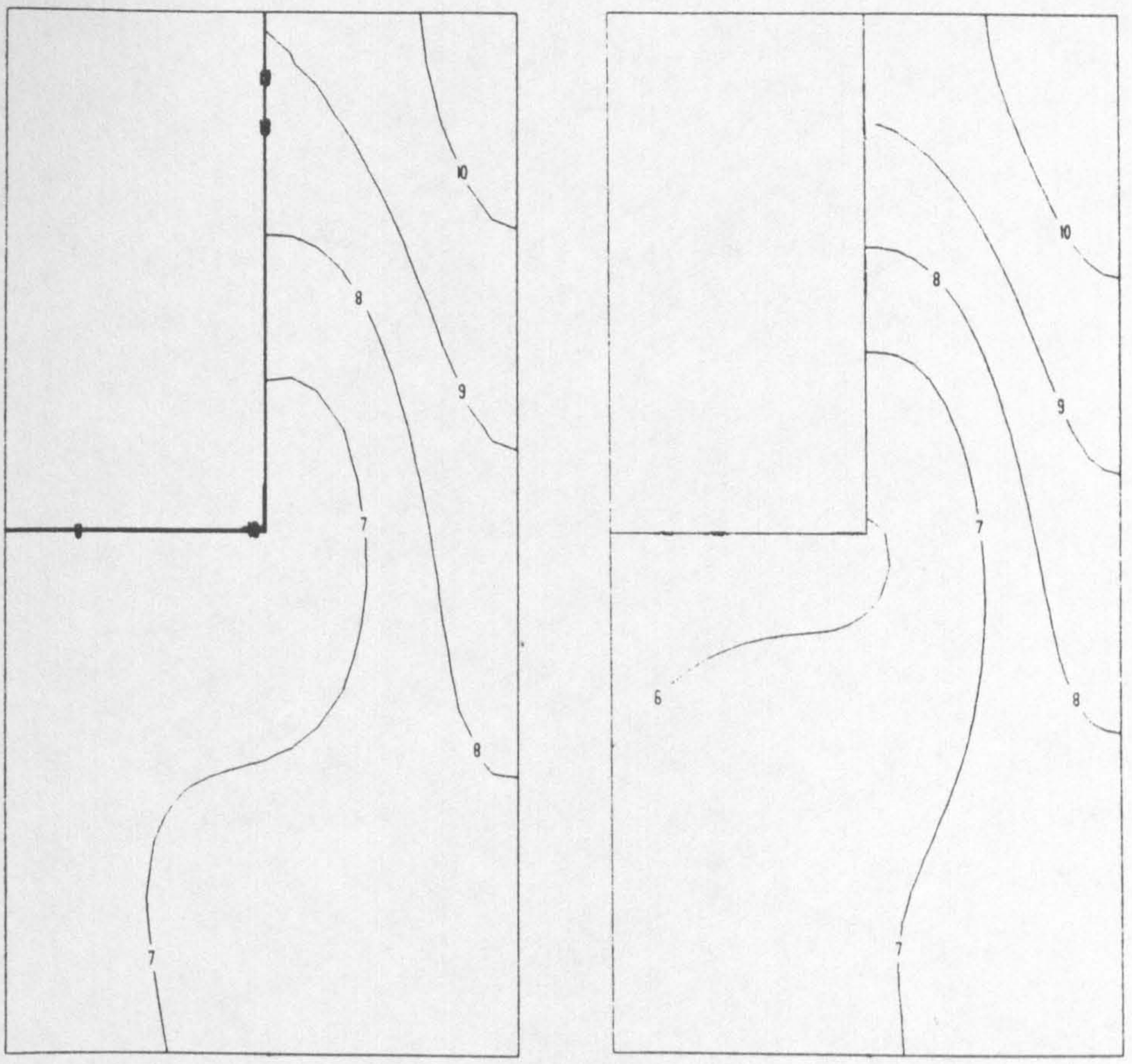
(a)



(b)

CONTOUR KEY	
1	0.100E+00
2	0.150E+00
3	0.200E+00
4	0.250E+00
5	0.300E+00
6	0.350E+00
7	0.400E+00
8	0.450E+00
9	0.500E+00
10	0.550E+00
11	0.600E+00
12	0.650E+00
13	0.700E+00
14	0.750E+00
15	0.800E+00
16	0.850E+00
17	0.900E+00

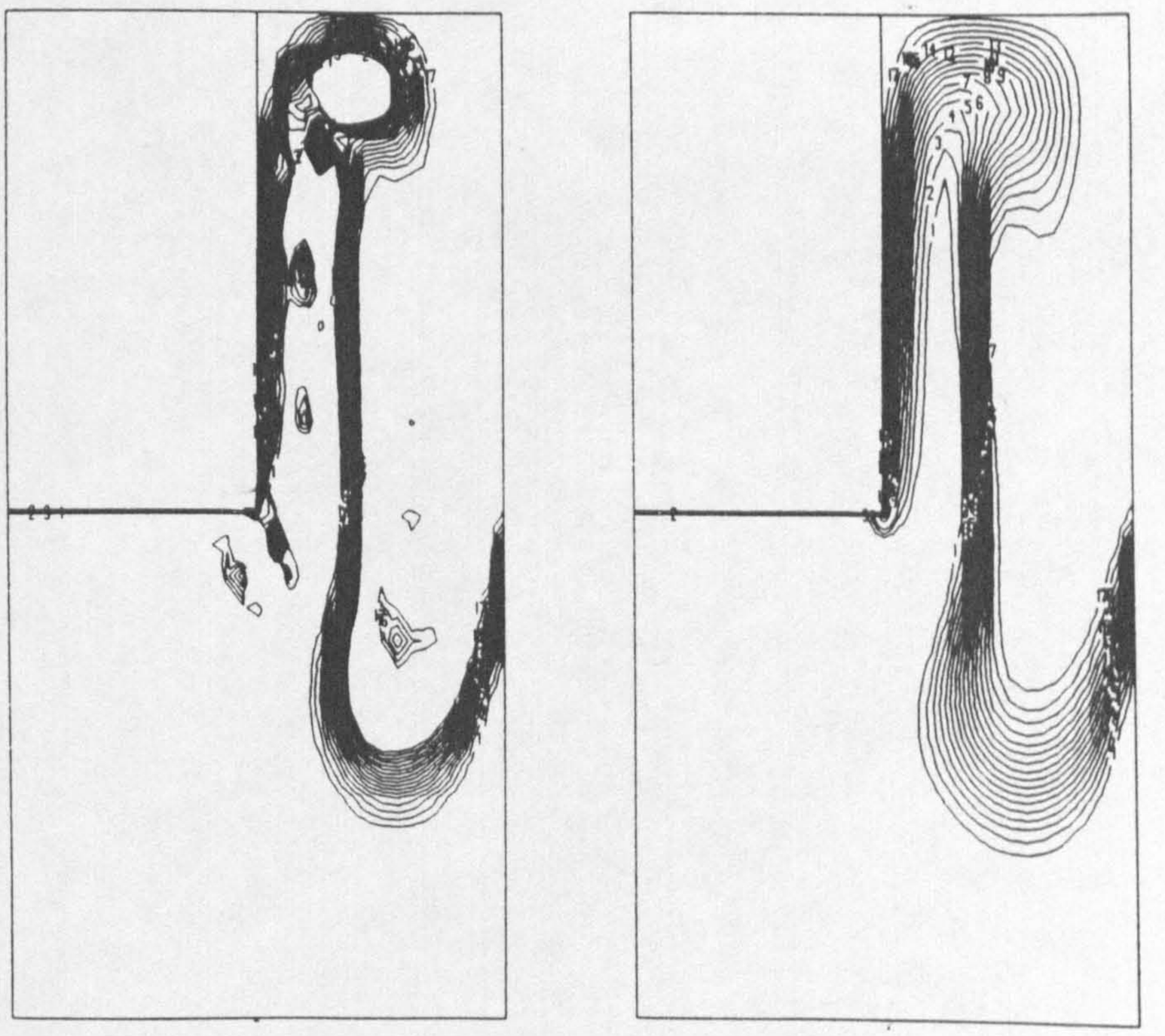
Figure 5.48 Concentration fields at Re=10 for a Newtonian fluid after 20 secs, produced by; (a) D.G. Knight using the finite difference method and (b) non-upwinded approach.



CONTOUR KEY	
1	0.100E+00
2	0.150E+00
3	0.200E+00
4	0.250E+00
5	0.300E+00
6	0.350E+00
7	0.400E+00
8	0.450E+00
9	0.500E+00
10	0.550E+00
11	0.600E+00
12	0.650E+00
13	0.700E+00
14	0.750E+00
15	0.800E+00
16	0.850E+00
17	0.900E+00

(a) (b)

Figure 5.49 Concentration fields at Re=10 for a Newtonian fluid on reaching a 75% mix, produced by; (a) D.G. Knight after 231.8 secs and (b) after 235.6 secs.



CONTOUR KEY	
1	0.100E+00
2	0.150E+00
3	0.200E+00
4	0.250E+00
5	0.300E+00
6	0.350E+00
7	0.400E+00
8	0.450E+00
9	0.500E+00
10	0.550E+00
11	0.600E+00
12	0.650E+00
13	0.700E+00
14	0.750E+00
15	0.800E+00
16	0.850E+00
17	0.900E+00

(a) (b)

Figure 5.50 Concentration fields at Re=100 for a Newtonian fluid after 10 secs, produced by; (a) D.G. Knight non-upwinded finite difference approach and (b) D.G. Knight upwinded finite difference approach.

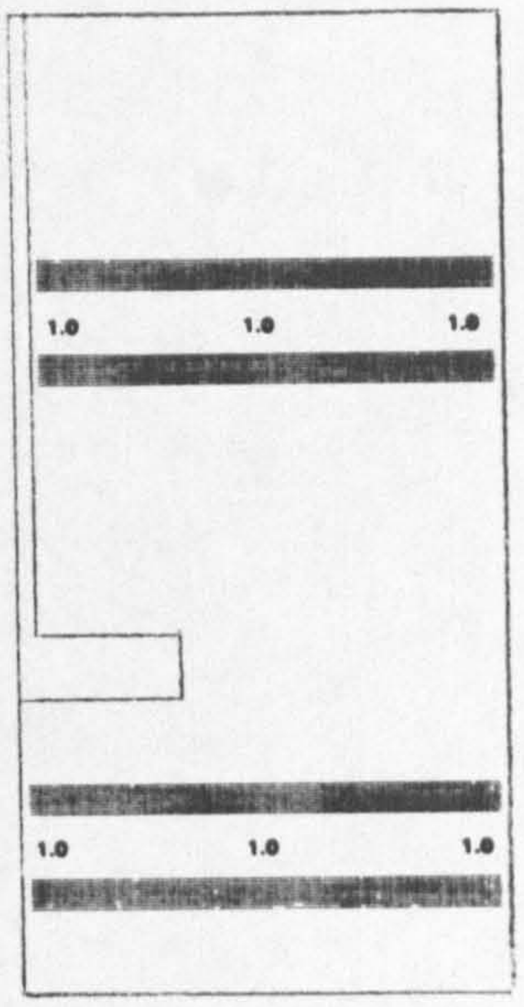
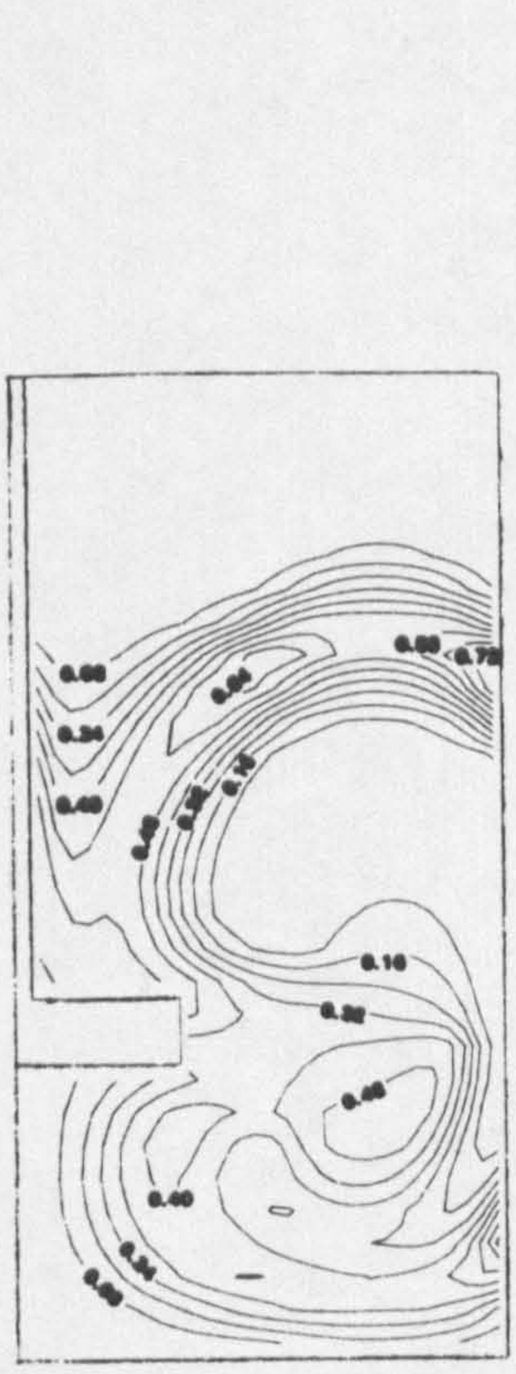
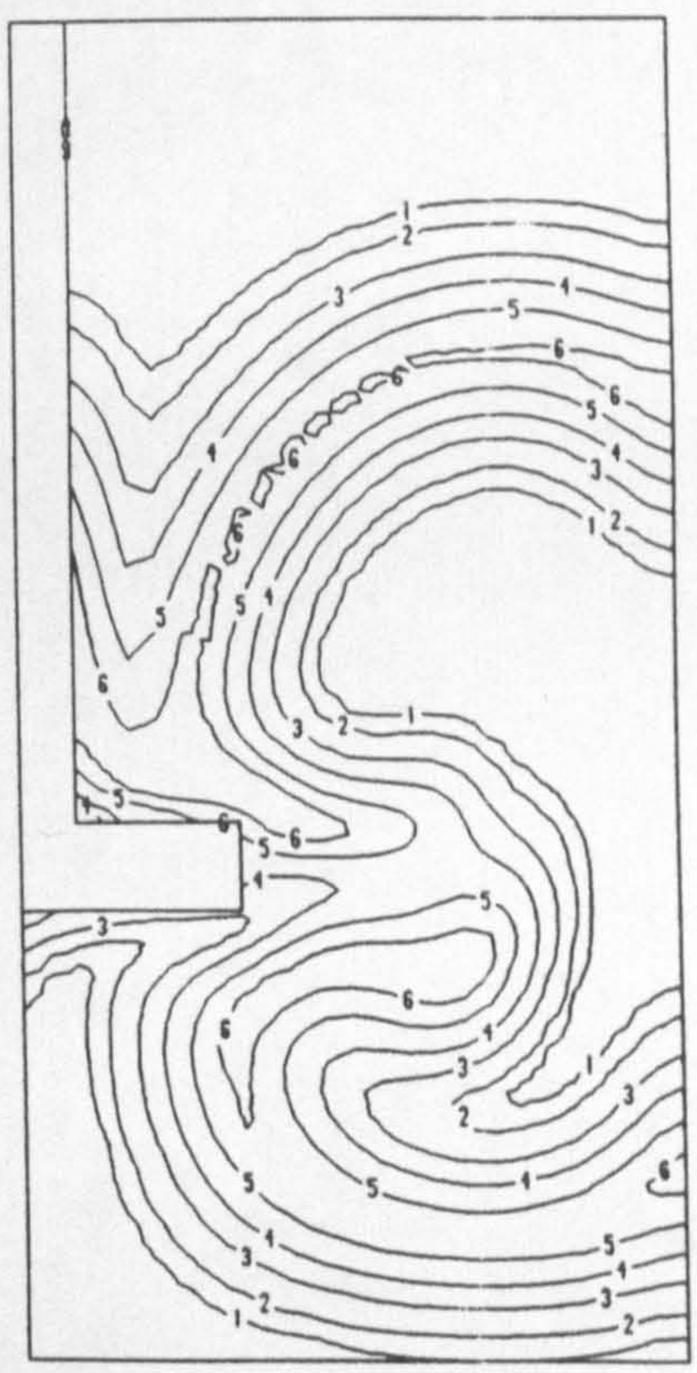


Figure 5.52 The geometry and initial conditions employed by Spragg et al.



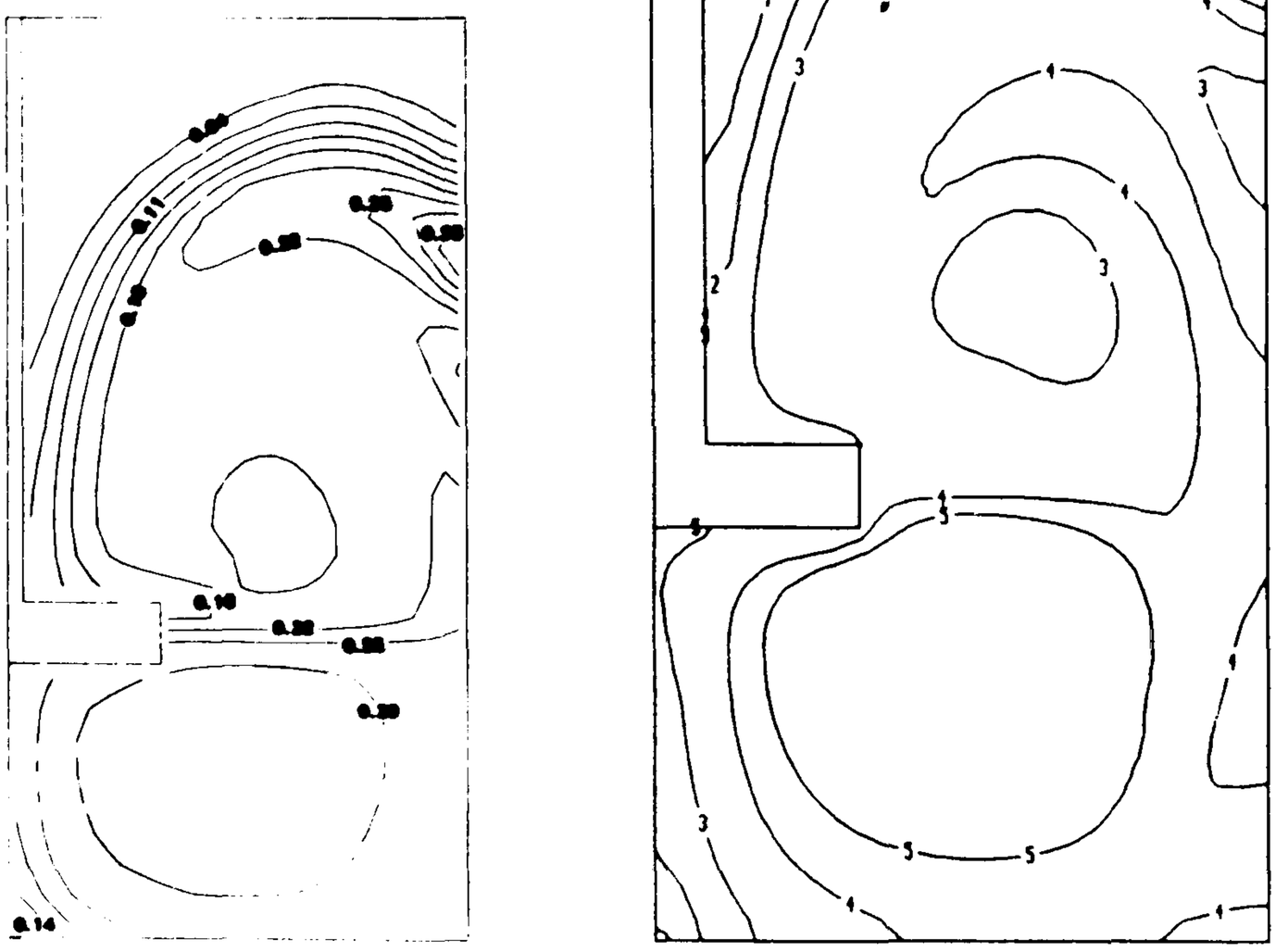
(a)



(b)

CONTOUR KEY	
1	0.100E-01
2	0.400E-01
3	0.160E+00
4	0.350E+00
5	0.600E+00
6	0.780E+00

Figure 5.53 Concentration fields for non-Newtonian flow; (a) by Spragg et al for Re=17 after 5 revs and (b) for Re=26.7 after 31.4 secs.



(a) (b)

CONTOUR KEY	
1	0.400E-01
2	0.900E-01
3	0.162E+00
4	0.270E+00
5	0.330E+00

Figure 5.54 Concentration fields for non-Newtonian flow;
(a) by Spragg et al for Re=17 after 20 revs and
(b) for Re=26.7 after 125.8 secs.

Chapter 6

The Development of a Multifrontal Finite Element scheme to solve Fluid Dynamics problems on MIMD Networks.

6.1 Introduction

The results presented in the previous chapters were obtained using a VAX 8650 multi-user super-mini computer. The finite element programs used were computationally intensive and therefore took a considerable amount of C.P.U. time to run - many hours in some cases. This computer is a multi-user machine so that the actual "turn-round" time (elapsed time) is very long and often extends to days. To continue solving these types of problems for more complex elastic fluids it is clearly necessary to find some way of substantially reducing the "turn-round" time whilst also attempting to decrease the C.P.U. time.

Firstly attention was centred on the algorithm which had been implemented, to see if in any way it could be made more efficient. Newton's method was implemented, as described in a previous chapter, and considerably reduced the number of iterations to reach convergence. Yet the only way to make a dramatic improvement to the "turn-round" time would be to transfer to a different computer, either a super-computer or a parallel architecture machine.

We had available to us, at the Rutherford Laboratory via a Janet link, the Cray-XMP; however, another option was open to us, that of working on a parallel architecture machine. The department has an IBM compatible PC fitted with a network of transputers.

This, with an efficient algorithm, could also produce a suitable speed-up and be a solution to our problem.

My supervisor Dr R.G. Miles was already developing parallel algorithms to solve finite element problems on a network of processors and therefore it seemed a natural progression for me to proceed in this direction. It was seen as a far greater challenge for me to develop parallel algorithms, grasping new concepts of working in parallel, than switching to work on a faster sequential machine. I also had the advantage of not entering uncharted waters as my supervisor had developed some Occam code for transputers, and had also developed a useful communications harness. Having the equipment on site and readily available were major advantages over liaising with an establishment many miles away with only a very limited amount of time allocation allowed for use of their machine.

For quite a while it has been perceived that computation speed can be significantly bettered by utilising parallelism. Due to the cheapness of the microprocessors and the ability to link large numbers of these microprocessors together in a loosely-bound network, this realisation has come to fruition. The design of algorithms to run on such a network and that will attain the network's full potential, is an area of research in which there is much current interest. The desired objective is for a program executing on a network of N processors to produce a speed-up in the order of N times that achieved on a single processor. In practice such speed up is seldom achieved due to communication, configuration and the lack of parallelism in the algorithm.

6.1.1 A survey of Finite Element based Parallel Schemes

A large number of researchers are currently involved in attempting to design efficient parallel algorithms for the finite element method. Most of the work so far has been tailored for the analysis of structures. A summary of some of the schemes recently suggested will be outlined. The first two papers mentioned relate to algorithms designed for fluid dynamics problems whilst the remainder relate to the analysis of structures.

Amestoy and Tilch [1] solved the compressible Navier Stokes equations for the Karman-Vortex-street problem adopting the finite element method. The multifrontal method was employed for the solution of a large set of linear equations produced by the above formulation and implemented on a shared memory multiprocessor system. In developing the optimised Taylor-Galerkin formulation of the problem it produced a diagonally dominant symmetric sparse matrix. They also applied a reordering scheme suggested by Markowitz [69] maintaining the sparsity of the matrix by reducing the amount of "fill in" , the scheme is known as the minimum degree algorithm.

Williams and Glowinski [124] manipulated irregular meshes on a distributed memory multiprocessor machine. They achieved load-balancing as the solution progresses instead of attempting to load-balance the mesh at the beginning. The algorithm also had a general finite element function which can deal with linear, multilinear and non-linear operators, one of these applications being an incompressible Navier-Stokes solver with a three stage implicit time step. The solution of a linear system of equations was achieved by a preconditioned conjugate gradient approach whilst Newton's method was adopted if the problem was non-linear.

Barragy and Carey [3] have developed an element-by-element solution scheme for elliptic boundary-value problems whereby element matrices and right hand sides are computed concurrently and stored in their global positions in separate individual overall stiffness matrices, which are obviously extremely sparse and so only the dense element contributions are stored. Boundary conditions are imposed at the element level which again is performed concurrently. At this stage the global RHS is formed from the individual RHS of each element.

No special partitioning of the domain or element ordering are required. The equations are solved by applying a preconditioned conjugate gradient approach, a parallel Jacobi preconditioner is used which has proved generally effective. For self-adjoint problems

with symmetric systems a conjugate gradient element-by-element scheme is used. This approach has been implemented on a shared memory multi-processor machine.

Farhat and Wilson [31] recommend substructuring the domain and describe an automatic procedure to do so for arbitrary shaped domains. Both direct and iterative solution schemes are mentioned. The parallel direct scheme proposed incorporates a concurrent LDL^T matrix decomposition as well as parallel forward and backward substitution. For the parallel iterative scheme a multicolouring successive over relaxation (SOR) scheme is suggested. They have implemented their ideas on a local memory multi-processor machine.

Nour-Omid et al [84] also adopted a substructuring of the mesh technique. To solve the problem they employed direct triangulation factorisation for the interior variables of the substructure followed by a preconditioned conjugate gradient method for the remaining variables which were on the substructure interfaces. They also implemented their scheme on a local memory multi-processor.

Melhem [70] suggested a technique which is based on ideas taken from both frontal methods and band matrix solvers. It is a customary practice when using the frontal method to order the elements but in this adaption it is necessary also for the nodes to be ordered so that they can be eliminated in sequential order. Here the assembly and factorisation can be performed concurrently and also for the special techniques available for matrix factorisation it is a requirement to have sequential ordering.

6.1.2 The Proposed Parallel Scheme

In this chapter a new approach is described, that of a general and efficient parallel algorithm based on the finite element method applied to fluid dynamics problems. The algorithm adopts the multifrontal scheme and is implemented on multiprocessor networks. For the algorithm to be efficient, the load on all of the processors in the

network should be balanced and the time taken to solve the problems on a workstation should be considerably reduced for a network of processors in comparison to that achieved by a single processor.

The algorithm is designed for a MIMD network, where MIMD is the abbreviation for a Multiple Instruction Multiple Data architecture. This type of architecture can be thought of as a number of processors which are linked together via communication links and each processor works independently. Processors have the ability to communicate with each other via their links through a routing network. Each processor is an independent unit with its own local memory, arithmetic unit and instruction counter.

The type of MIMD networks that we have considered have the following properties :

(i) each processor has its own local memory of 1M byte or more and the memory bandwidth of the network increases in proportion to the number of processors present in the network;

(ii) each processor is controlled by its own set of instructions although it is often the case that a large number of the processors will be running the same process on different data sets;

(iii) communication between processors in a network is achieved by the simple operation of message passing;

(iv) the composition of the network is adaptable for an arbitrary number of processors, whether it is a large or small system;

(v) there is no shared memory and no shared clock between processors, which makes each processor a totally independent unit which communicates with other processors by message passing.

When designing an algorithm it is important to think in parallel; it may not be possible to take a sequential algorithm and convert it into a parallel one. When designing algorithms it is important that a very high percentage of the work can be performed in parallel and the instances of processes being run sequentially in the algorithm are minimised. The algorithm should also be general enough to be implemented on an arbitrary sized network. There are a number of features which attribute to the design of a good algorithm:

(i) The granularity of the problem. A fine grained problem has many processors working on small amounts of data with a lot of communication taking place between processors, which can cause bottlenecks to occur and also much time wasted in communication. For large grained problems the processors are given work to do independently on large sections of data with the amount of communication between processors kept to a minimum.

(ii) A simple organisational data structure is required so that throughout the algorithm each processor knows exactly from where it is to receive or to send information and is in the correct state to do so.

(iii) The speed at which the scheme works is dependent upon how efficiently each processor is used. Therefore, it is important to balance the work load over each processor in the network. If the amount of work given to one of the processors greatly exceeds that on the other processors, then this processor may delay the other processors, causing them to become idle waiting for information, and in so doing will effectively control the network.

6.2 Frontal and Multifrontal Schemes.

6.2.1 The Frontal Method

The frontal scheme was described briefly in chapter 4 and for coherence is described in greater detail here. The frontal method is a modified form of Gauss Elimination and can be split into three distinct stages:

(a) Assembly of the Element Matrix.

(b) Elimination of fully summed variables.

(c) Back-Substitution.

In the frontal method the elimination stage begins to take place as soon as possible long before the system matrix is fully assembled. This results in a considerable saving in computation time and memory space because work is being carried out on relatively small matrices in comparison to manipulating the full system matrix.

We next give a brief description of the three stages mentioned above.

(a) **Assembly stage.** The procedure begins by assembling the first element and incorporating it into the system matrix. The assembling of the elements and the adding of their nodal contributions into the system matrix continues until the system matrix reaches a pre-defined critical value. The number of nodal variables in the system matrix is known as the frontwidth. At this stage, provided the frontwidth has been chosen correctly, there should be a number of nodal variables which are fully summed and therefore some of these variables can be eliminated, to allow another element to be assembled.

(b) **Elimination stage.** In the method described here the pivots are chosen from the diagonal terms for the fully contributed nodal variables with the selected pivot being the variable having greatest absolute magnitude. When a pivot has been chosen

the pivotal row is then stored, with its associated variables and right-hand side. The row is then eliminated from the system matrix and the matrix compressed.

Steps (a) and (b) are continued until all of the elements have been assembled, then step (b) is continued until the elimination of the remaining variables is complete.

(c) **Back-Substitution.** Finally the back-substitution is performed returning the values for the variables defined at the nodes.

The method is summarised diagramaticaly in fig. 6.1.

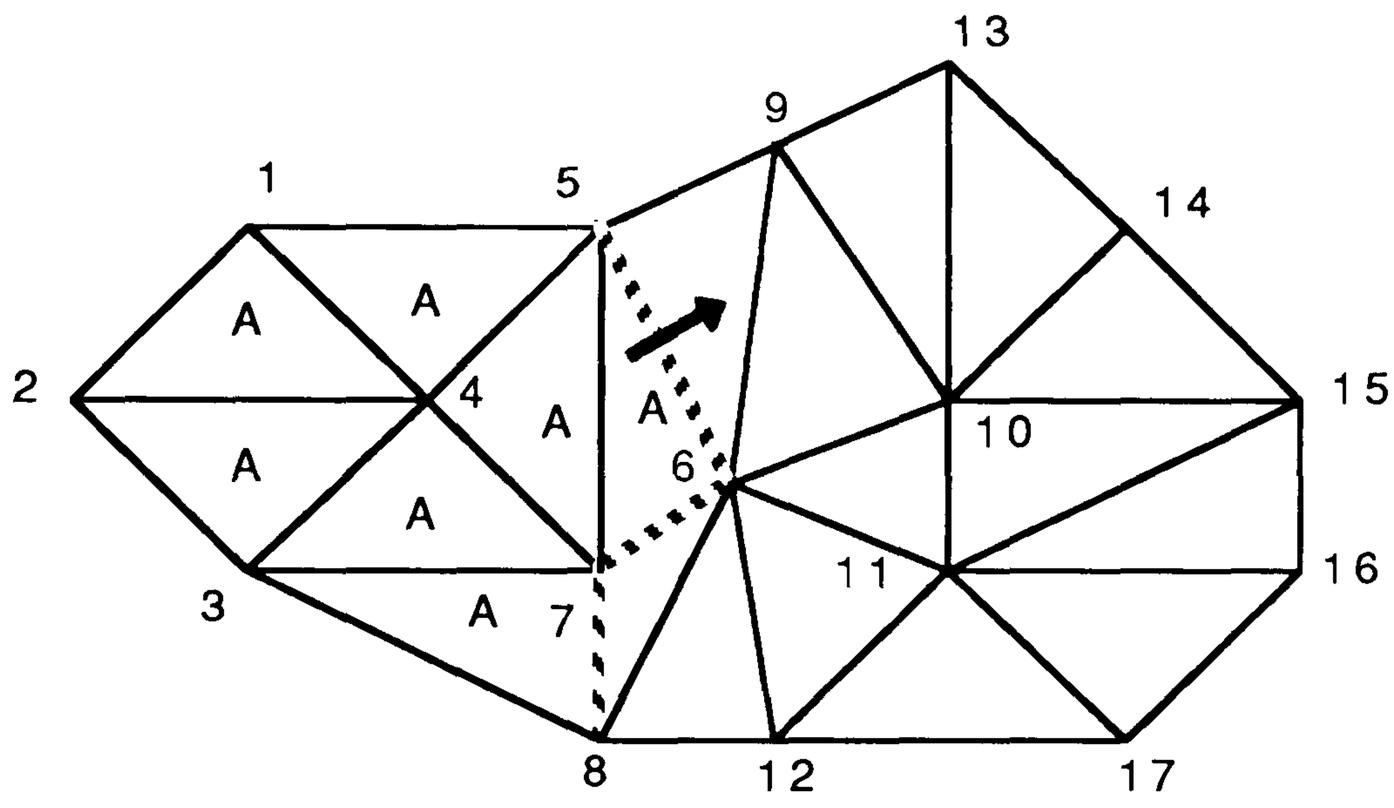


Figure 6.1 A mesh of triangular elements with a single front.

The diagram represents a mesh of three noded triangular elements, with one degree of freedom at each node. The dashed line signifies the current position of the front as it moves across the mesh in the direction of the arrow. From the diagram it can be seen that a point in the procedure has been reached where seven elements have been assembled, these elements are indicated by the letter A in the diagram, and the contributions to nodes 1 to 4 are fully specified and the rows associated with them in the system matrix are complete. At this stage the rows associated with nodes 1 to 4 have been stored and eliminated from the matrix. The nodes remaining in the front are nodes 5 to 8 which are

the active nodes, and are dependent on contributions from elements yet to be assembled. The nodes behind the front have been eliminated and those ahead have yet to enter the system.

6.2.2 The Multifrontal Method.

The multifrontal method of Reid [98] is an improvement to the frontal method. He suggests substructuring the finite element mesh and uses a binary tree data structure to represent the order of assembly. In the multifrontal method instead of having only one front moving across the domain there are a number of fronts which are started from different parts of the boundary. The way in which the method works is best illustrated visually (fig. 6.2).

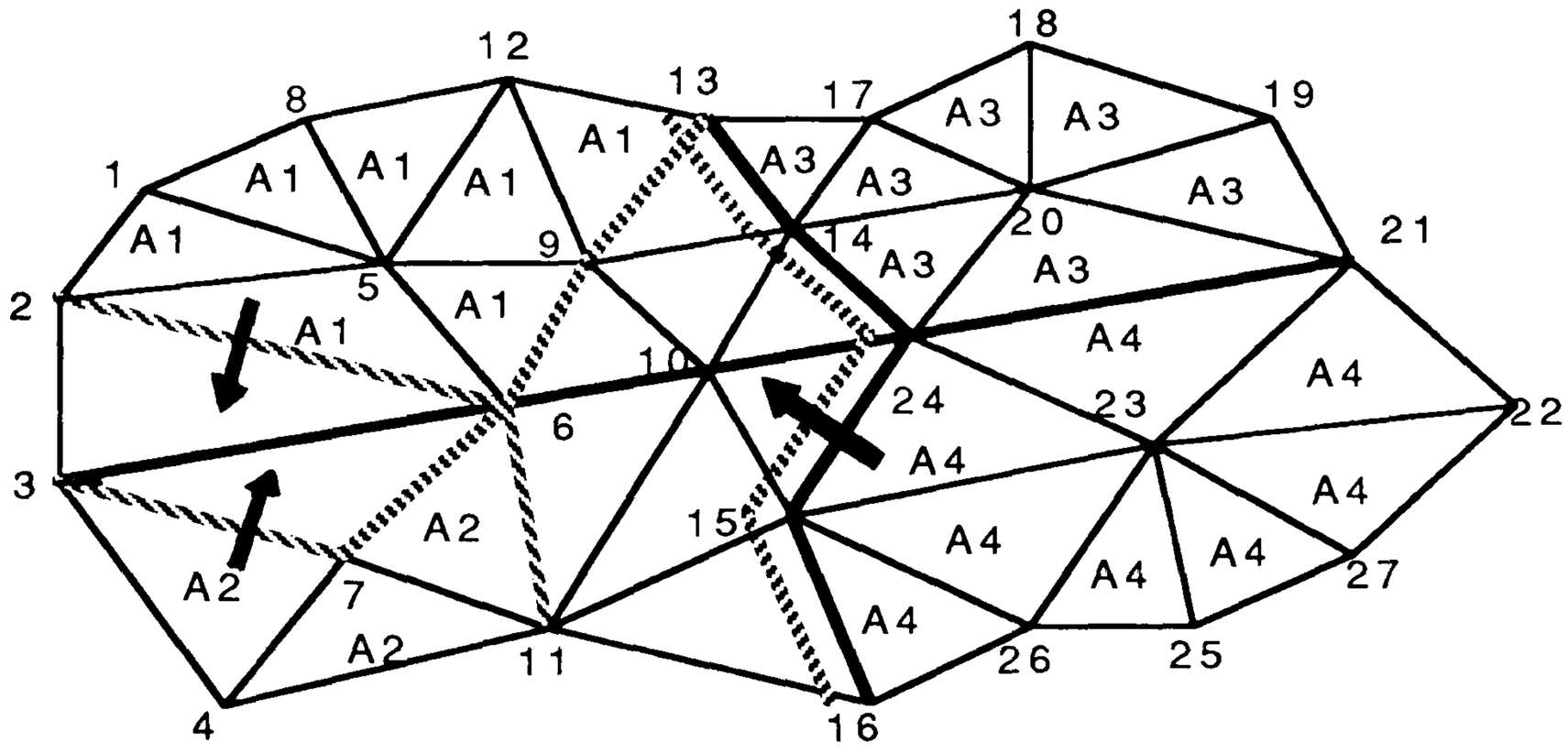


Figure 6.2 A mesh of triangular elements partitoned into four sub-domains

Here the assembled elements are signified by Ai with i indicates the sub-domain to which they belong. The thick black line indicates the sub-domain interfaces and the dotted line represents the current position of the fronts.

The domain has been substructured into four smaller sub-domains, with four fronts having been started, one in each sub-domain. From the figure it is seen that the four fronts have moved across their individual sub-domains assembling a number of elements and eliminating a number of variables. In sub-domain 3 and 4 all of the elements have

been assembled and all interior variables eliminated i.e. for sub-domain 3 variables 17,18,19 and 20 have been eliminated and for sub-domain 4 variables 22,23,25,26 and 27 have been eliminated. For sub-domains 3 and 4 we are left with fronts which contain those variables which lie on the sub-domain interfaces, i.e. sub-domain 3 has variables 13,14,24,21 remaining and sub-domain 4 has variables 16,15,24,21 still remaining.

The order in which sub-domains are combined is specified by the tree data structure shown in (fig. 6.3), where the tree is traversed from the leaves to the root for the elimination process.

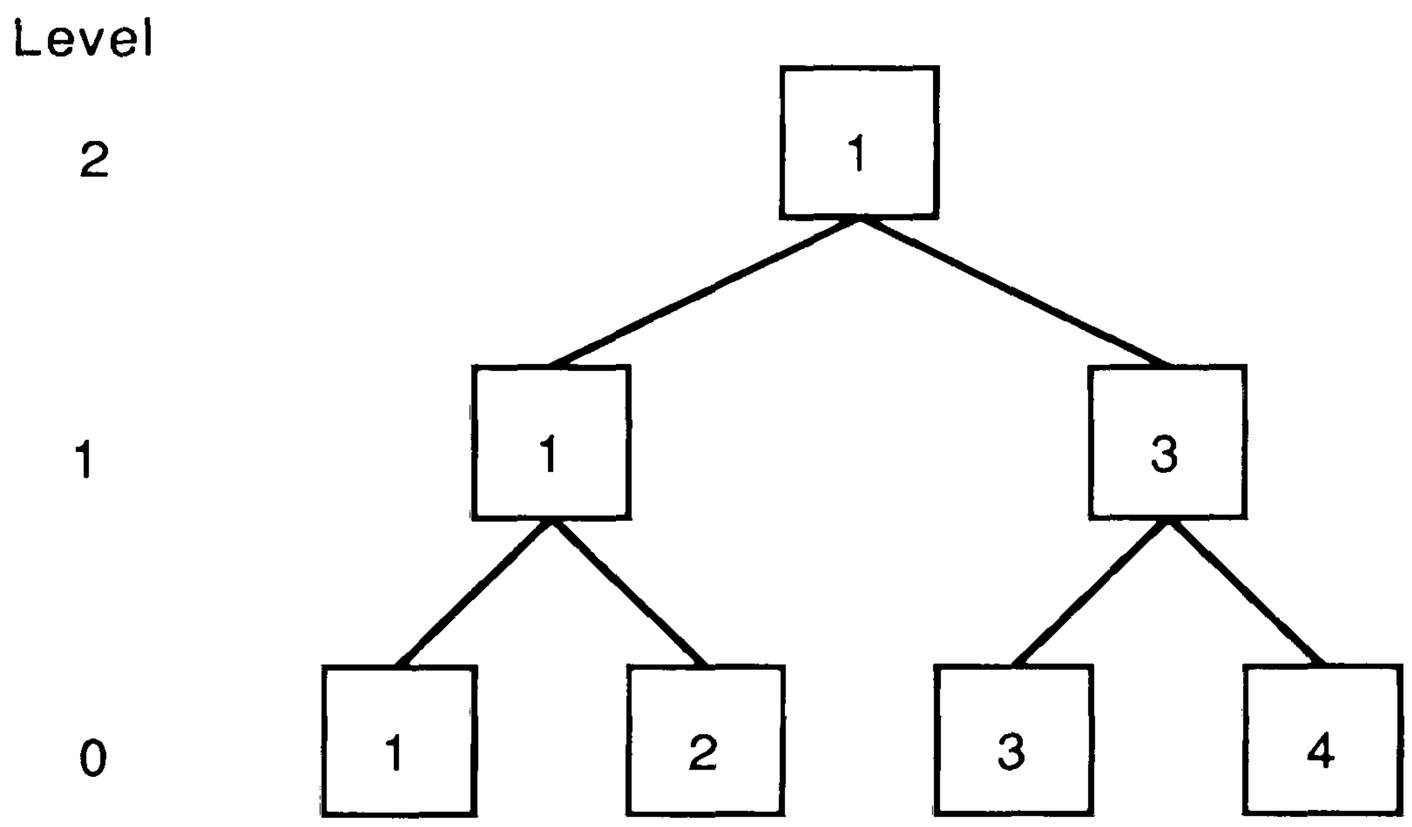


Figure 6.3

From the data structure sub-domains 3 and 4 are to be combined. These two sub-domains are coalesced merging the two fronts together allowing for fully summed variables, which are common to both, to be eliminated, i.e. for sub-domains 3 and 4 only variable 21 is eliminated. This procedure is continued until all of the sub-domains have been coalesced to form larger sub-domains, specified by level 1 of the tree. The same process is repeated on these larger sub-domains, level 2 of the tree, until all sub-domains have been combined. The combination of sub-domains is defined by the organisation specified in the tree data structure and the number of times the process has to be repeated to complete the coalescing stage is calculated from the number of levels in the tree, the value is defined as the number of levels minus one i.e. coalescing is required at two levels of the tree, level 1

and 2. When the top level of the tree is reached all of the remaining variables are fully summed and can be eliminated and then the back-substitution performed.

It is clearly seen that the multifrontal scheme has a large amount of inherent parallelism. When considering the scheme in greater detail, it was noticed that the back substitution stage could also be done in parallel. The binary tree which was traversed from the leaves to the root for the elimination stage, could be traversed in the opposite direction, from the root to the leaves, for the back-substitution stage.

6.3 Transputer Systems

6.3.1 Hardware

The transputer can be thought of as a complete computer in silicon and is built by INMOS Ltd. The derivation of its name "transputer" is an amalgamation of two words '**trans**istor' and 'com**puter**'. Each of these chips contains a processor, memory and communication links. We have used two types of transputer:

(i) The T414 which has a 32-bit microprocessor, 2K bytes of on-chip RAM, an external memory interface, and four standard communication links. All of the floating point operations are done via software.

(ii) The specification of the T800 is similar to the T414 except that it has a 64-bit floating point processor and 4K bytes of RAM, both on chip. It is also pin-compatible with the T414. A schematic diagram of the T800 transputer is shown in fig.(6.4).

The four communication links allow communication between transputers on a point-to-point basis, and in this way networks of arbitrary sizes and topologies may be

constructed. Since the links and the processor are largely independent, the processor can continue working whilst the transputer is passing messages via the links. Each link enables synchronous bidirectional communication relating to two channels, one of the channels being an input and the other an output.

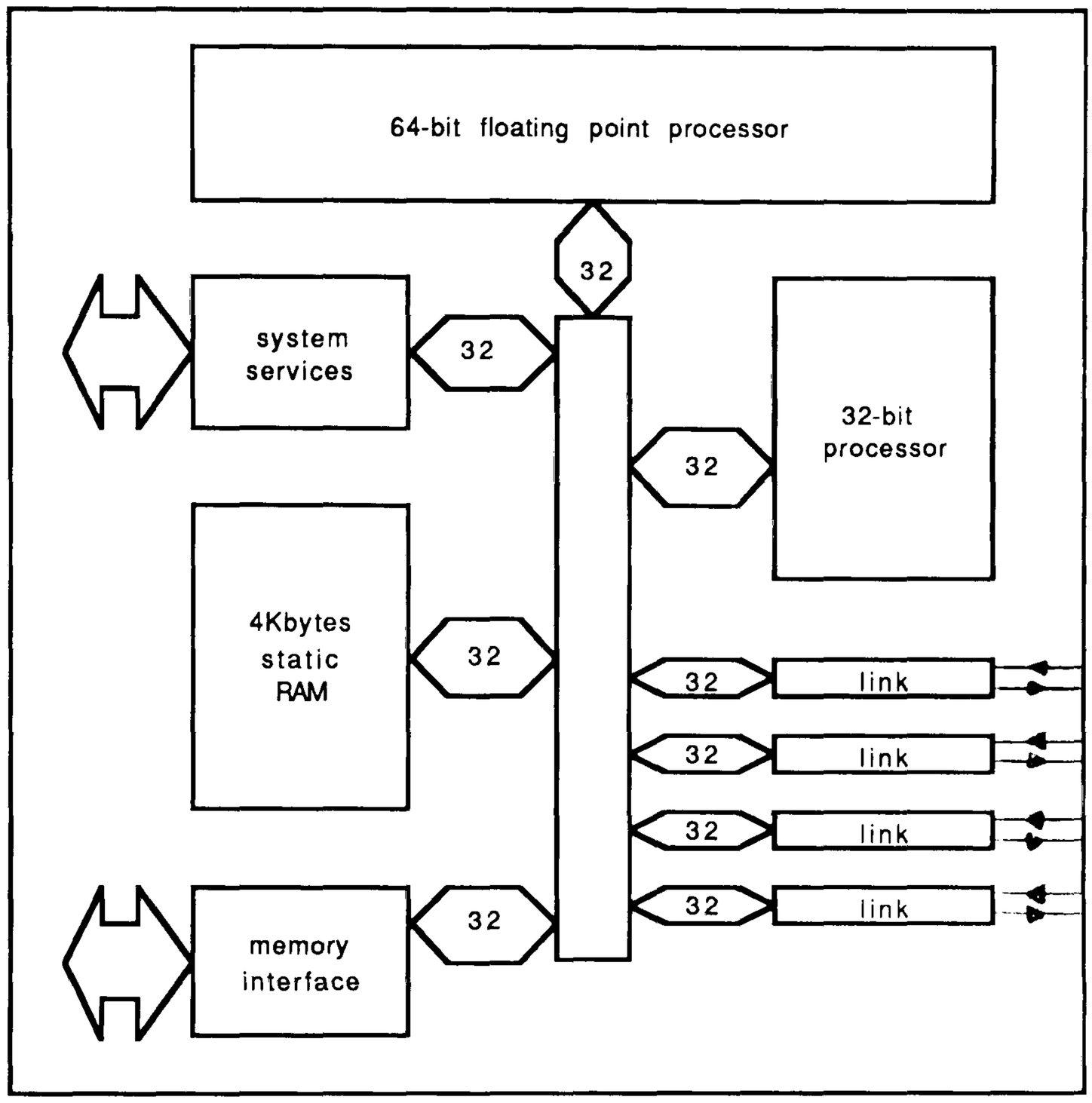


Figure 6.4 Schematic diagram of a T800 transputer

6.3.2 **Occam**

The Occam programming language was written jointly by C. Hoare of Oxford University and D. May of Inmos Ltd. specifically for the transputer. They took its name from the 14th century philosopher William of Occam known for his principle 'entities should not be multiplied beyond necessity' or 'keep it simple'. The entire code we have written for the transputer is in Occam as it is still the best language for the transputer even though by now many compilers have been written in other languages such as Fortran, Pascal and C. Occam is considered as the assembly language of the transputer and is also classified as a high level language, which was developed from a computational model which incorporates communication, parallel execution and synchronisation in its structure and exploits the concurrency of the transputer very efficiently. Occam is founded on the principle of

communicating sequential processes, which allows sequentially run processes to be combined with other such processes and for these seperate processes to be run in parallel.

In Occam processes communicate by message passing via channels with specific protocols. A channel defines a single direction coupling of two concurrent processes. Communication on a channel is synchronised, i.e. for two processes connected by a channel, communication will only occur when both processes are in a state of readiness and messages are transmitted as a sequence of bytes. When a data byte is transmitted, the sending transputer waits to accept an acknowledgement which tells it that the receiving transputer is ready for the next byte. This guarantees reliable transmission in spite of any possible delays in either transputer involved.

An Occam program can be run on an arbitrary number of transputers and it is interesting to note that it is possible to run the same program on a reduced number of transputers, or on a single transputer with little or no modification having to be made to the program. The communication specification of an Occam channel is independent of any specific hardware implementation. The configuration used specifies the way in which an Occam program is loaded on particular hardware and couples particular processes with actual processors and also specific channels with actual hard links.

Occam programs are composed of a few simple primitive processes, which can be coupled together to form larger and more complex processes using special Occam constructs, three of these being SEQ, PAR and ALT. The SEQ construct, where SEQ is an abbreviation of the word sequential, stipulates that the constituent processes inside the SEQ are to be executed sequentially, one after another finishing on the completion of the final constituent process. This construct resembles the Pascal construct BEGIN and END. A simple example is presented of the construct:

```
INT num.a,num.b:
SEQ
  channel.in ? num.a
  process.a
  channel.out ! num.b
```

This section of code inputs an integer value from channel.in to num.a, performs process.a, where process.a signifies a number of simple assignment operations, and then outputs num.b to channel.out, with the channels having been declared earlier in the program. For Occam to be able to distinguish which processes are part of the SEQ, all of the constitutive processes of the SEQ are indented with respect to the SEQ by two spaces.

The PAR construct, where PAR is an abbreviation for parallel, stipulates that the constitutive processes of the PAR begin execution at the same time, terminating when every constitutive process has reached its completion. Care should be taken when using the PAR construct, following is a summary of few helpful guidelines. It is essential to remember that the concurrent processes of the construct are considered as independent processes, which means that the variables used in each concurrent process are local to that process and the only way communication between such processes can take place is via unidirectional channels, where only two processes can share a channel.

A simple example of the PAR construct is shown below:

```
CHAN channel.comm:
PAR
  REAL32 num.a, num.b:
  SEQ
    channel.in ? num.a
    process.a
    channel.comm !.b
  REAL32 num.c, num.d:
  SEQ
    channel.comm ? num.c
    process.b
    channel.out ! num.d
```

Here channel.in and channel.out have been declared in other processes. The first component process, process.a, of the PAR performs some simple arithmetic on the input variable before passing it on to the second component process of the PAR which again

performs a simple calculation on the input variable, process.b, before sending it out via the output channel.

The ALT construct, where ALT is an abbreviation for alternative, is a very important and useful construct, for example it is used in the writing of multiplexors. It enables certain decisions to be carried out subject to the current condition of channels. A simple example is presented to explain how the ALT can be used. Consider four predefined channels:

```
INT num:
ALT
   channel.a ? num
     process.a
   channel.b ? num
     process.b
   channel.c ? num
     process.c
   channel.d ? num
     process.d
```

In this example the ALT inspects the condition of each channel waiting for an input to be received. Control is given to the first channel which receives an input and subsequently its related process is executed and is the only process to be enacted.

6.3.3 Networks

The correct choice of network configuration can have a significant effect on the speed of execution. The transmission of data between processing elements should traverse the shortest path that connects the two processors in the network. This can be achieved by applying graph theory to determine the best configuration of the network and an efficient routing algorithm.

Inmos are currently constructing a range of boards, the most common for small systems has four transputers hard-wired in a square. These building blocks are useful for constructing transputer networks. The boards that we have used are composed of four T800 transputers each with 1M byte of memory attached and two of the links on each are wired to adjacent transputers, however, eight links remain unconnected. For the network

to be able to communicate with the outside world another single transputer is required as the root transputer, a T414 transputer with 2Mbytes of memory is used for this purpose, which links with the host computer. The most common small system host computer is the IBM-PC, where the boards fit into the expansion slots of the PC; other hosts are available e.g. the SUN. Fig. 6.5 is a schematic diagram of a simple 4-transputer network, which has a root r to the host.

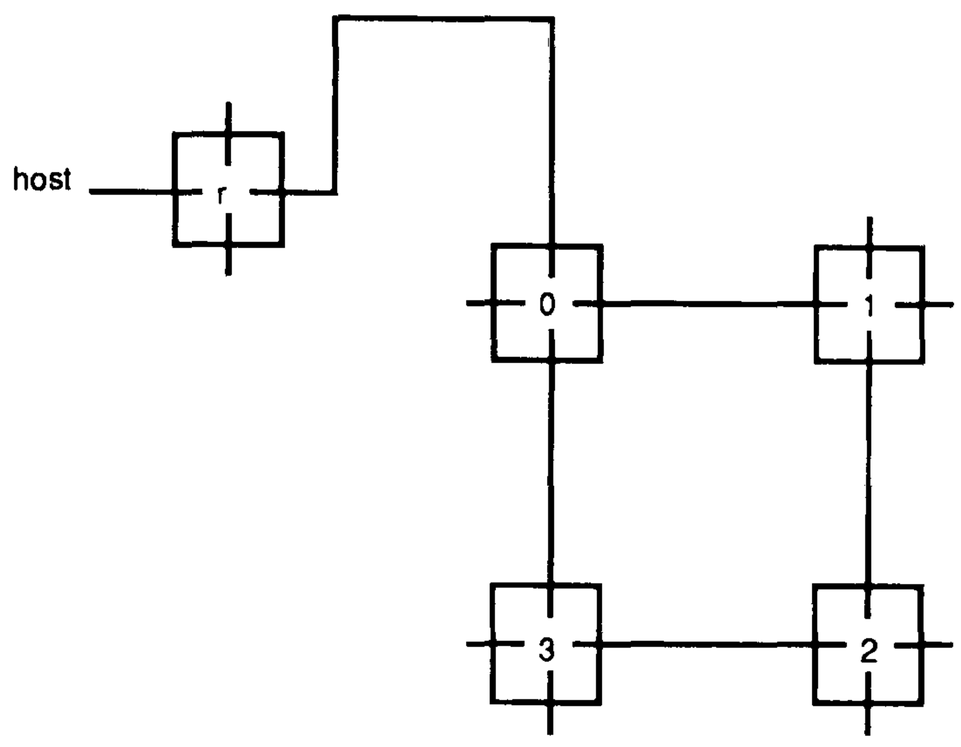


Figure 6.5

To connect a greater number of transputers into the network, more 4-transputer boards can be inserted into the expansion slots. These are then daisy chained together using the unconnected links to produce the desired configuration. There are many ways of configuring the system, such as a ring, array or modified hypercube.

To produce the results presented in this chapter we networked four, eight and sixteen transputers in a bi-directional ring configuration, which is a very straightforward configuration for routing messages in either the clockwise or anti-clockwise directions. The aim is to enable the messages to get to the required processors in an efficient and reliable manner. Fig. 6.6 shows a schematic diagram of a configuration of 8 transputers in a ring with the root r to the host, and a graphics board s linked to the root.

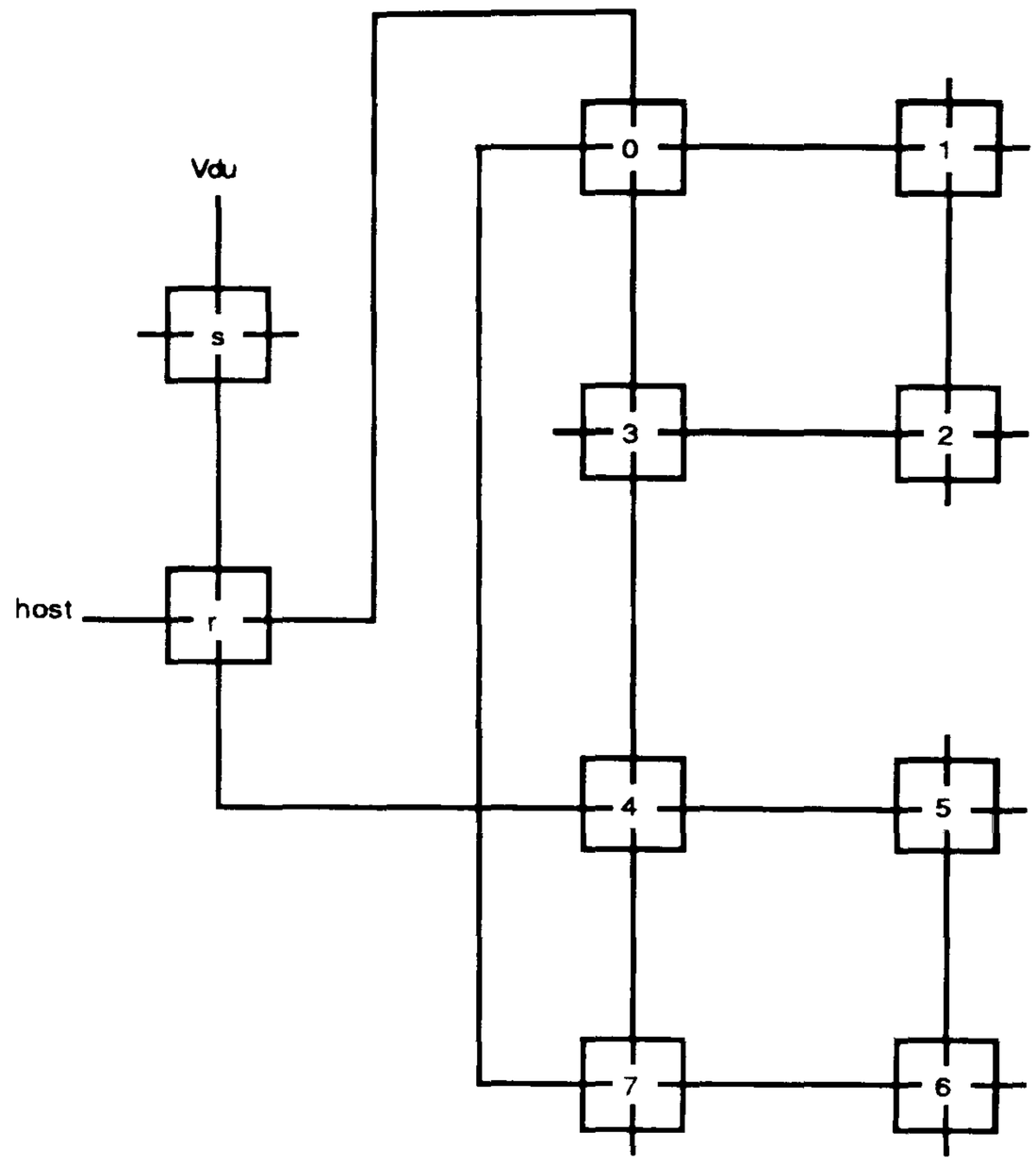


Figure 6.6

Networks with this type of configuration are only useful if there is a very small amount of communication or if the communication is with the nearest neighbour, which is the case for the parallel algorithm explained in the next chapter. For example with sixteen transputers, the diameter of the network is eight, however, a diameter of four is achieved by using a two-dimensional array. For more details on configurations there is available an Inmos technical note on configurations,Hill [38] and a paper by Saad and Schultz [101].

Chapter 7

Parallel Application to 2-D and 3-D axisymmetric flow problems

7.1 Design specifications

It was specified in the introductory section of the previous chapter that when designing a parallel algorithm three factors had to be considered namely, granularity, load balancing and an organisational structure for communication. We shall deal with these aspects in this section.

Firstly the domain is sub-divided into a number of sizeable partitions and specific parts placed on each processor in the network. Loaded on each processor is a modified frontal method which is applied to each partition, thus giving a large grained problem, and as a significant percentage of processor time is spent working on its own partition the communication between processors is minimised.

The organisational structure for communication is achieved by using a binary tree (fig. 7.1).

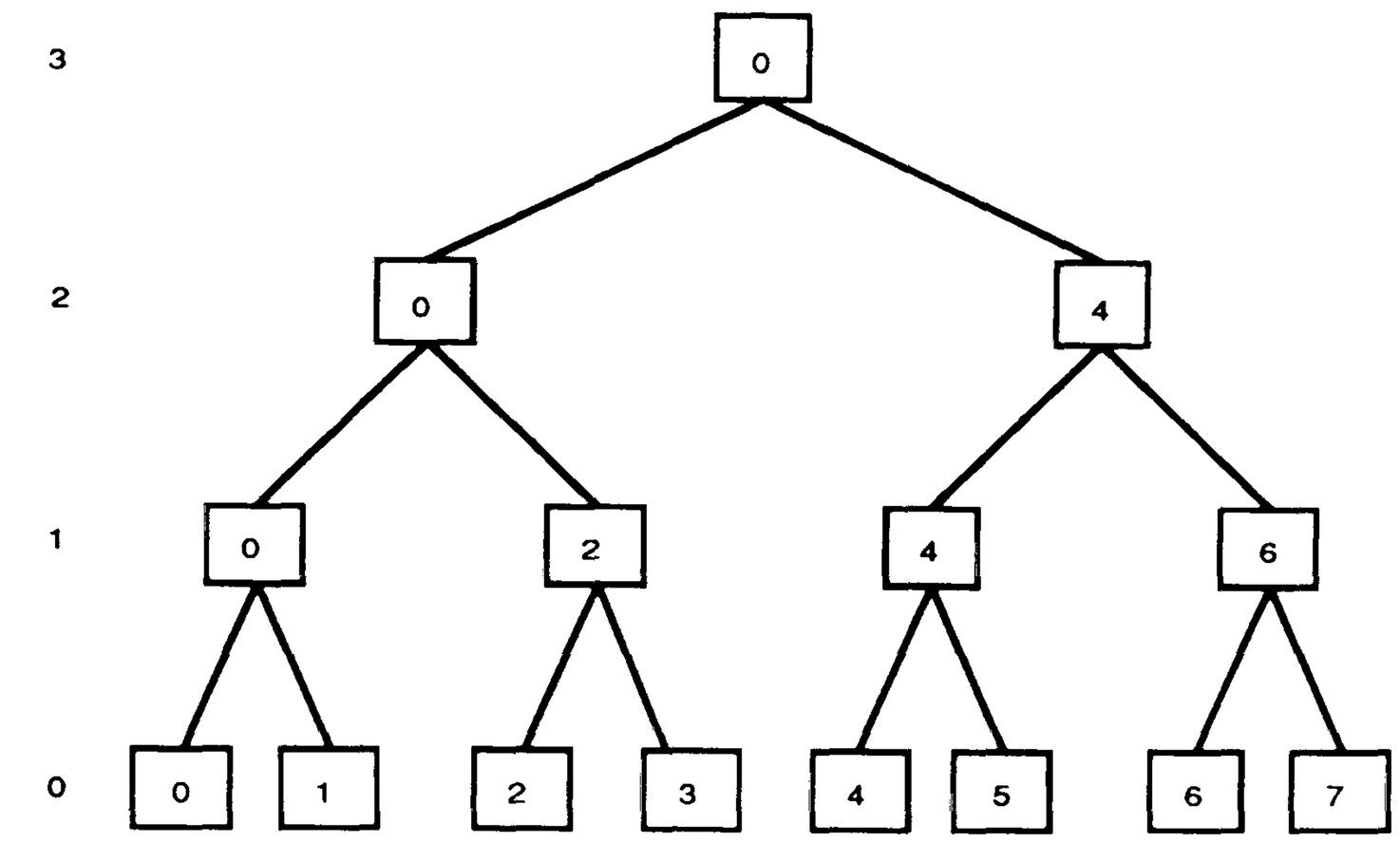


Figure 7.1

Each processor in the network has knowledge of the tree structure and from it knows whether it is to send or receive information and will be in a suitable state to do so. Initially the tree is traversed from its leaves to the root for the elimination stage, then in reverse order for the back-substitution. In Fig. 7.1 the numbers in the boxes represent processor numbers and indicate which processors are to receive information and from whom. By inspecting the tree it appears that at levels exceeding 0 a number of the processors may become idle. To overcome this a procedure is adopted for the elimination stage whereby the processors which become idle act as slaves, thus improving the efficiency of the eliminations at these levels, (see Fig. 7.2).

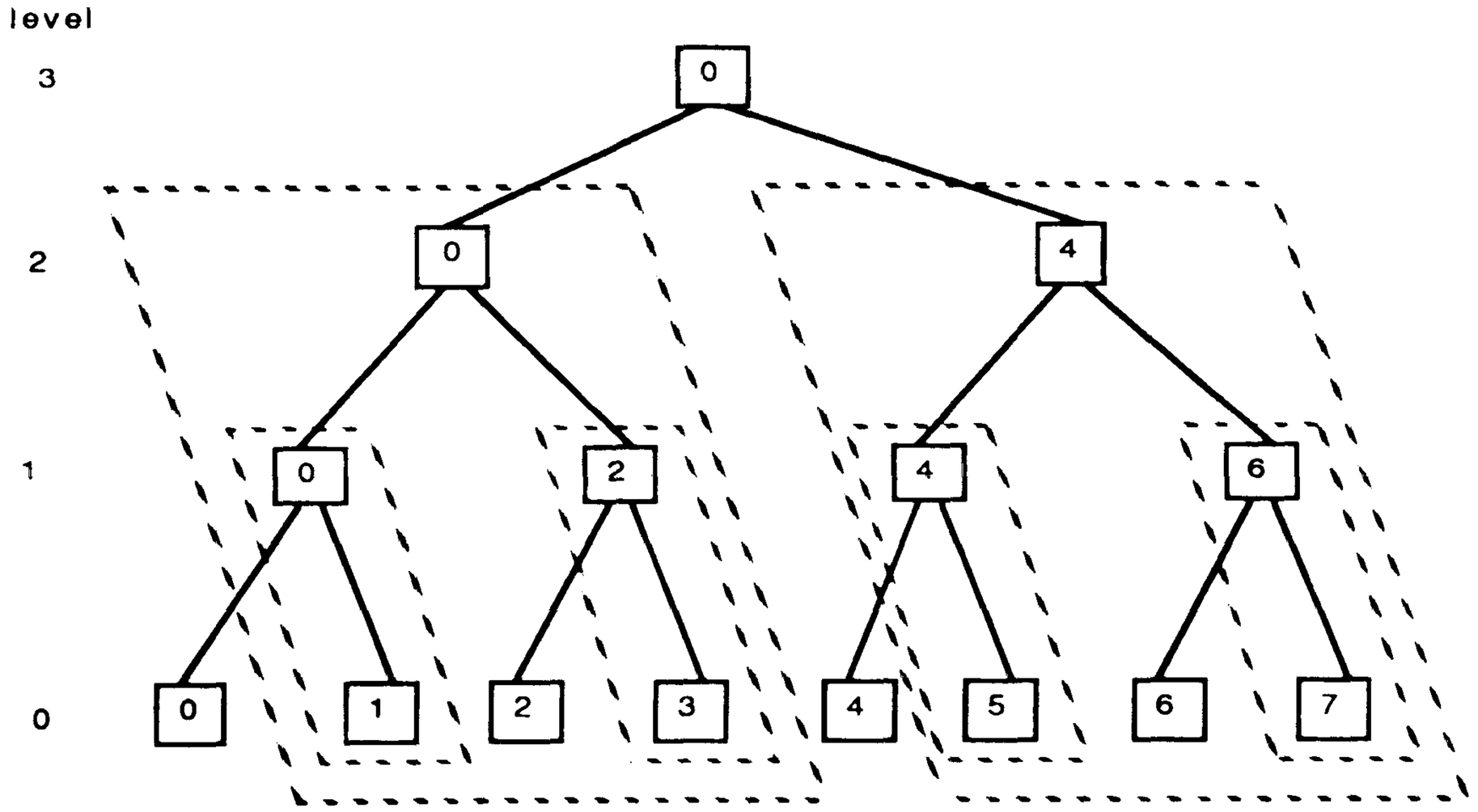


Figure 7.2

At level 0 of the tree each processor is working independently on its own sector. On completion, the odd numbered processors of figure (7.2), P_{2j+1} (j=0.....3), send their resulting frontal matrix to P_{2j}. This local root processor combines the matrices and if the resulting matrix is sufficiently large, with sufficient rows to be eliminated, it will allocate work to its slave processors. In fig. 7.2 the slave processors associated with a root are indicated by dashed line blocks. The same principle is applied to the other levels of the tree except for the top level. In general if we have n processors at level t of the tree $(0 < t < \log_2 n)$ processor P_{i2^t} $(i=0..... (n/2^{(t-1)}))$ receives information from $P_{(i2^t+2^{t-1})}$ and uses some of the slave processors $P_{(i2^t+j)}$ $(j=1.....2^t-1)$ to assist in the elimination.

This sharing of work in the elimination process for higher levels of the tree is best explained at level 1 with, say, processors P_0 and P_1 working on adjacent sectors of the finite element mesh. When the two processors have finished the level 0 work, P_1 sends its frontal matrix for sector 1 to P_0. Then P_0 coalesces the received matrix with its own and this results in a number of interface variables being fully summed and these are then available for elimination. Each of the rows associated with these variables must be eliminated from all of the other rows. At this level two processors are available so the local root processor, P_0, forms a packet of all the fully summed rows and half of the other rows and sends it to P_1. Now these two processors work independently on the elimination. On completion, P_1 returns its half of the matrix to P_0 which together with that on P_0 forms the complete matrix. There is some necessary duplication of work in this process, however a considerable time saving may be achieved.

This method can be adopted at all levels of the tree except the top level. The local root processor decides how many slaves to use depending on the size of the matrix and the number of rows to be eliminated. At the top level all rows are now available for elimination and so very much more communication is required if more than one processor is used, consequently we have not coded a shared process for this level. We have looked at the time taken in that part of the elimination phase where processor communication is

involved, i.e. at levels exceeding 0, and we have found by experience that up to a 50% saving in time is achieved by using the above approach rather than having no sharing of resources. At each level the number of eliminated rows is stored for use in the back-substitution.

To explain the way in which we overcame the difficulty of load balancing two problems are considered, (i) problems with a convex domain and (ii) problems with a non-convex domain. A domain is said to be convex if any two points of the domain can be connected by a straight line which lies completely inside the domain.

7.2 A Problem with a convex domain

7.2.1 Load balancing

Consider for example a driven flow over a square cavity. To achieve load balancing we require each processor to perform approximately the same amount of work. With a large grained approach load balancing can become a problem, for example consider fig.7.3.

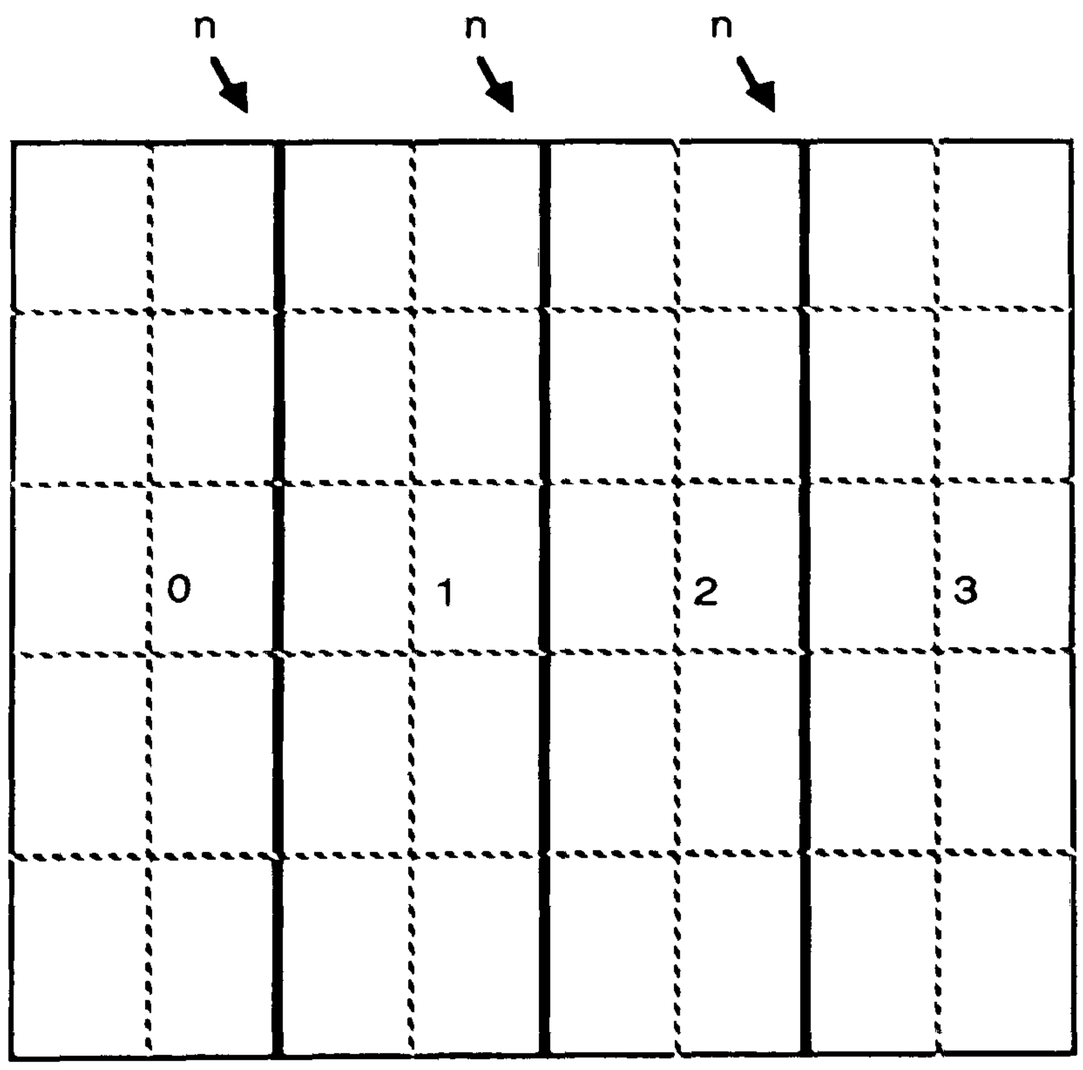


Figure 7.3

The figure shows a square cavity which has been partitioned into four equivalent parts. Assume that each partition is covered by a similar mesh composed of the same number of

elements and along the interfaces between partitions there are n variables. When the level 0 eliminations have been performed processors 0 and 3 will be left with fronts measuring n whilst processors 1 and 2 will have fronts of 2n, which will produce a significant difference in processor times, leading to a very poor load balancing.

An attempt may be made to balance this by placing a greater number of elements in the two end partitions, which would lead to refinement in a part of the mesh where it may not be necessary, therefore it is concluded that this is not an attractive way of splitting up the domain.

A simple solution to this problem which guarantees load balancing is to use a hub and spokes, see fig. 7.4.

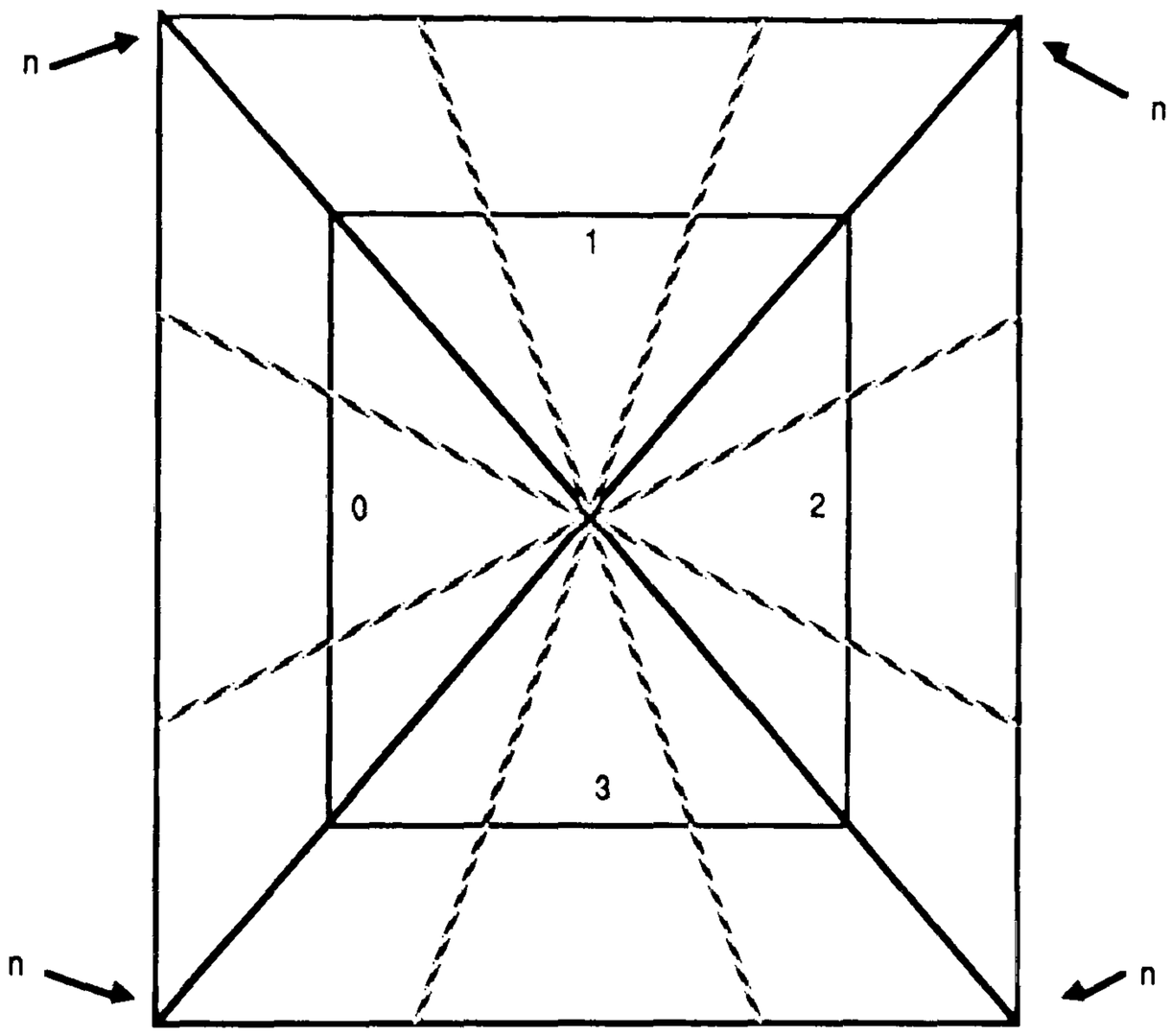


Figure 7.4

Consequently the domain is substructured in this way and a mesh formed within each partition. There has to be the same number of rows in each partition to ensure consistency across partition interfaces and also equal number of spokes in each partition to give the same number of elements in each sector. This domain ensures a balanced load and minimises the size of the frontal matrix throughout the process.

7.2.2 Sector ordering

The ordering and placing of partitions is also important and adjoining partitions should be placed on adjacent processors. This is best explained by an example. From the organisational tree, fig.7.1, it can be seen that the information produced by processors 0 and 1 are to be combined. If the partition placed on processor 0 relates to a distant part of the domain with respect to the partition placed on processor 1, then on coalescing the two they would not have any common fully summed variables to eliminate, causing the front to be unnecessarily increased.

The simple solution is to number the sub-region data for each processor in a clock-wise manner, i.e. the sub-region for processor 0 is adjacent to that on processor 1. When coalescing two sub-regions a number of common variables could be eliminated making the combined front equal to what it was before the coalescing had taken place. Therefore throughout the coalescing stage the overall size of the front will remain the same, except at the top level when all variables can be eliminated.

7.2.3 Mesh Refinement

This design of mesh is also good for mesh refinement and there are three types of refinement which can be carried out:

(a) A general refinement of the mesh can be achieved by increasing the number of spokes and rows as necessary, see fig. 7.5.

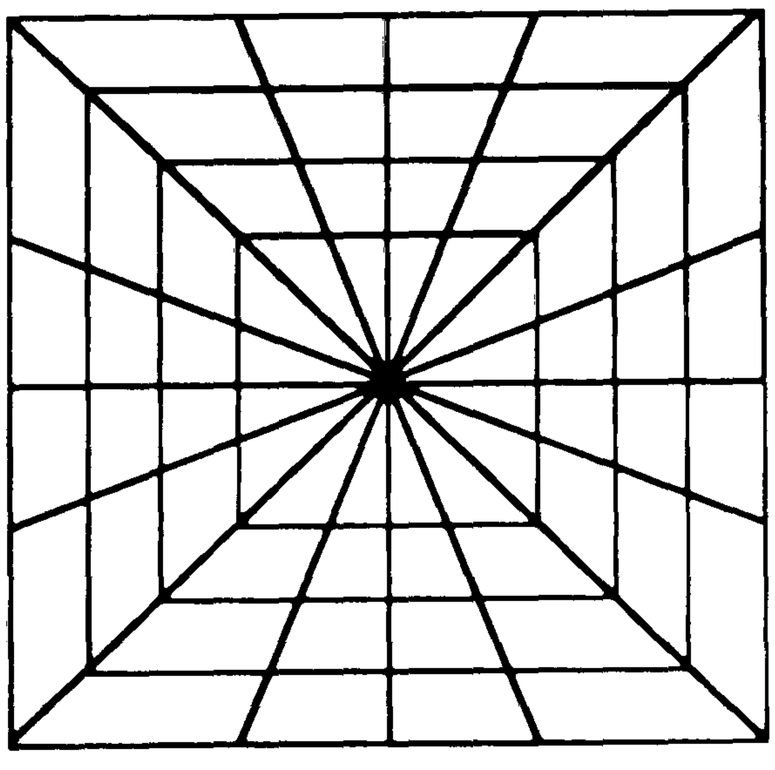


Figure 7.5

(b) Refinement at particular parts of the mesh, i.e. near the centre or close to the boundary of the domain. This is achieved by increasing the number of rows in these areas, see fig. 7.6.

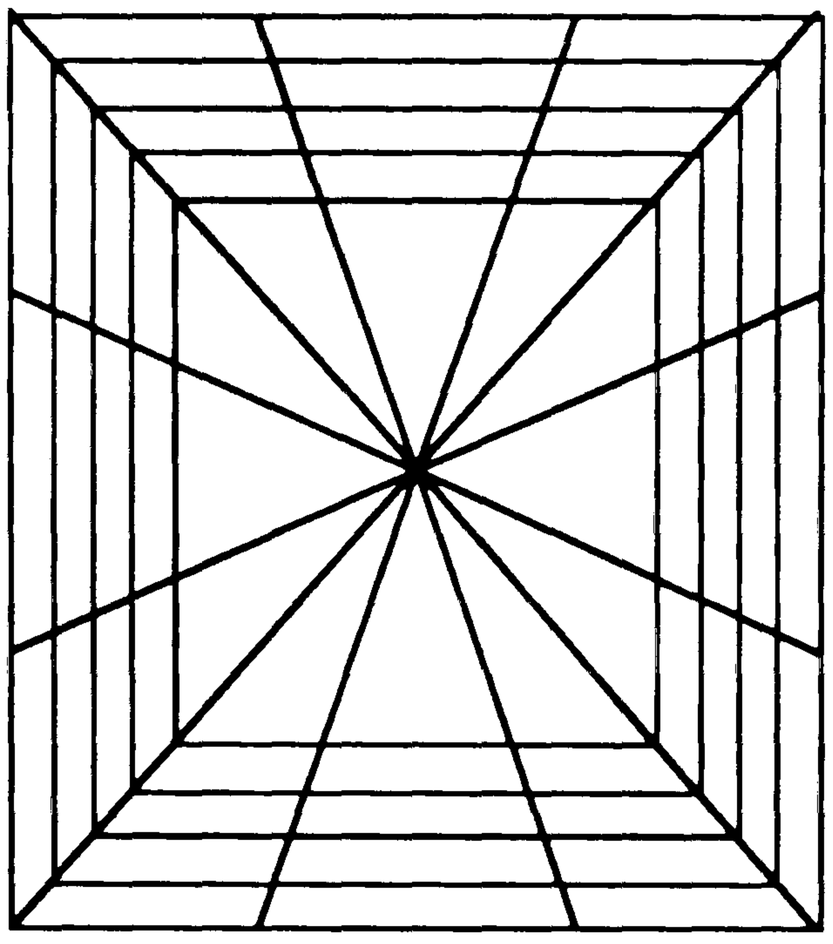


Figure 7.6

(c) Refinement at a specific part of the domain, i.e. three-quarters of the way up the domain. This is done by moving the centre point of the domain to the required position which gives a greater refinement close to the top of the domain, see fig. 7.7.

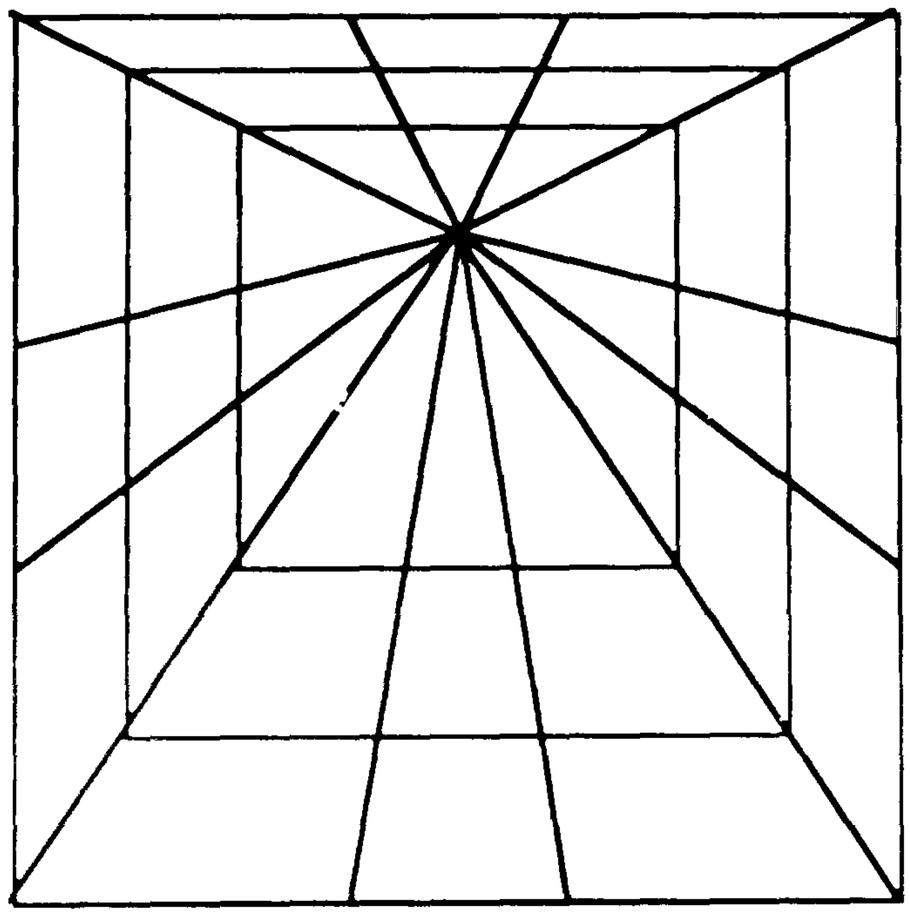


Figure 7.7

7.3 Problems with a non-convex domain

The procedure that has been applied to non-convex domains is to divide the domain up into the union of convex sub-domains with the methodology described in the previous section being adopted in each sub-domain. Each sub-domain is substructured into a number of sub-regions known as sectors. To explain the procedure consider the domain for axisymmetric mixing using a disc rotor partitioned into eight sectors as shown in fig.7.8.

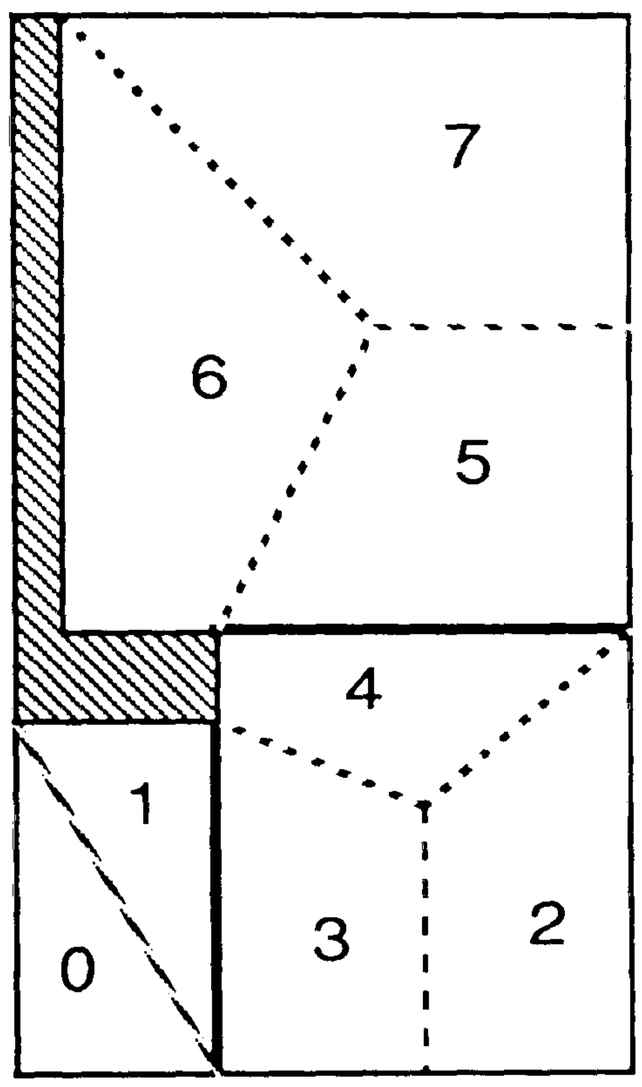


Figure 7.8

The domain is subdivided into three convex sub-domains, and each sub-domain is partitioned by a hub and spokes. In the example there are two inter domain interfaces, indicated by a thick line, and eight inter sector interfaces, indicated by a dashed line. There are eight sectors in all and therefore the organisational tree is as in fig.7.1.

For our application the geometry of a sector has to resemble one of two shapes; either a single triangular segment (fig.7.9) or two adjoining triangular segments (fig.7.10).

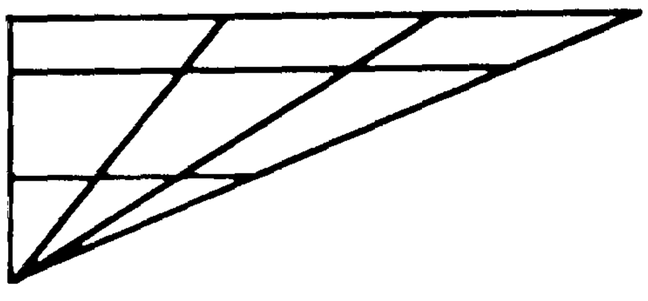


Figure 7.9

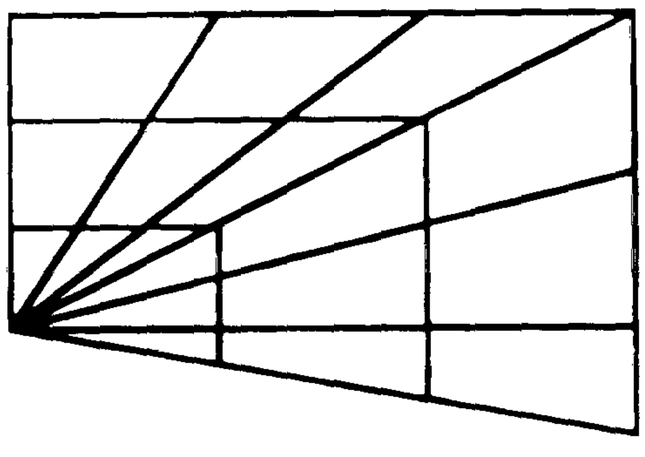


Figure 7.10

7.3.1 Load balancing

To balance the load for a non-convex domain is not as simple as that for the convex domain since not all the sectors are the same, some being on inter domain interfaces while others are not. To achieve an exact load balancing in general is very difficult. For this reason we want to develop a load balancing strategy that will give a balanced load, so that in the elimination phase the difference in time between the fastest and slowest processors is a small percentage of the total time to complete the stage.

Firstly it is essential to identify the major factors which are responsible for the work done in a sector. There are three main factors, namely the frontwidth, the number of variables and the number of elements. For problems of significant size the time for manipulation of the system matrix greatly exceeds the time spent in element assembly and on that premise the latter can be neglected. There are two parameters which can be varied, the number of spokes and the number of rows in a sector. To accomplish balancing at levels greater than 0 in the elimination the number of rows has to be fixed throughout the domain. Therefore the only parameter which can be varied in a sector is the number of spokes.

An approximation to the work load on a sector is given by the product (frontwidth)*(number of variables). The frontal method on each processor has a variable frontwidth, therefore we shall specify an average frontwidth in the approximation. Expressions for both these terms can be found in terms of the number of spokes and rows in a sector for our application. Firstly the number of rows and spokes are specified on the most constrained sector and from this information the load factor in each sector may be calculated and the number of spokes may then be adjusted so that a balance of the load on each processor can be achieved.

The way in which we choose to divide up the domain means that four different types of sector can arise. These are dependent on how the domain has been partitioned and are categorised in the following way:

Case A:

This sector defined by two segments and one interdomain interface.

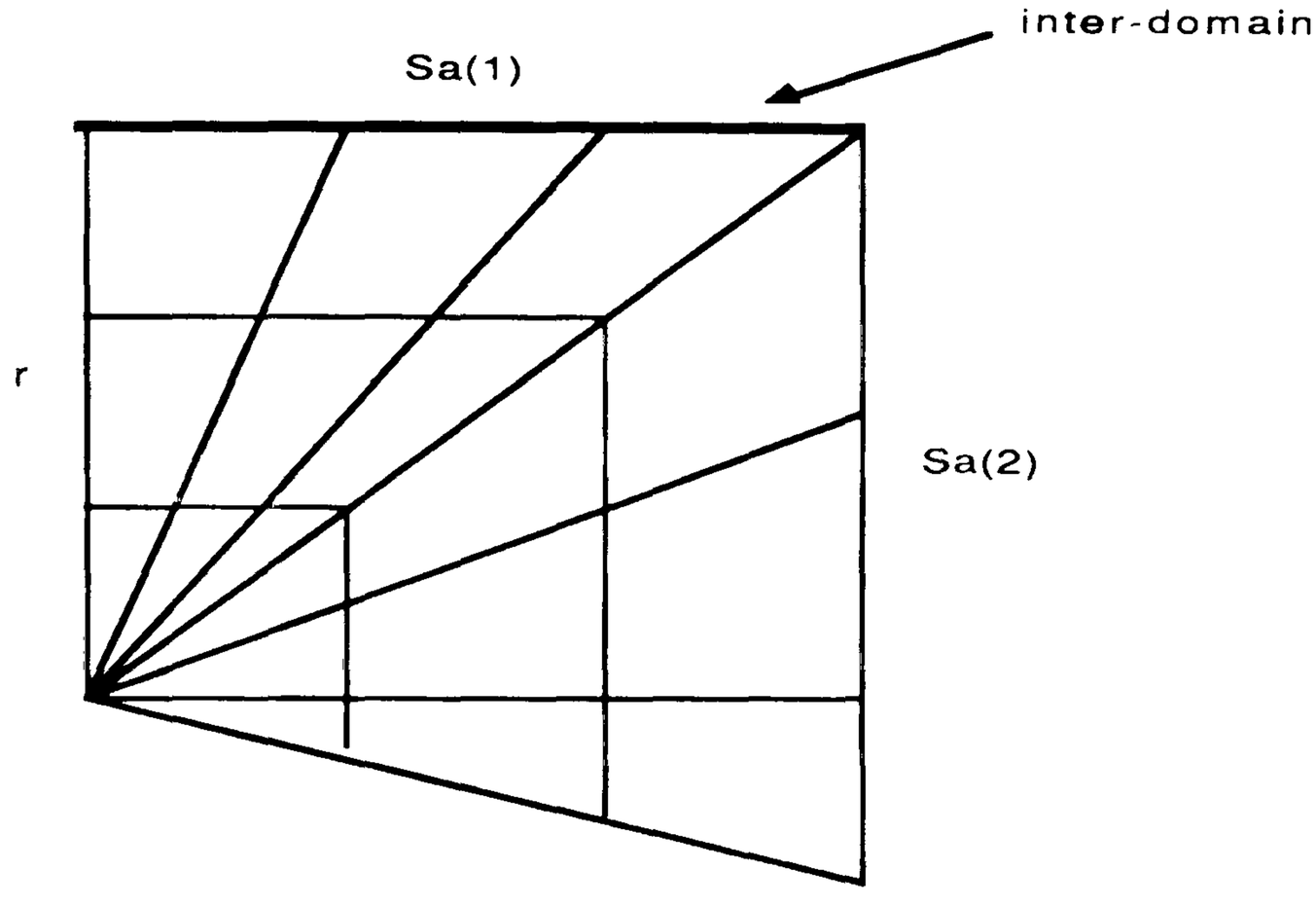


Figure 7.11

Here Sa(1) and Sa(2) signify the number of spokes in their respective segments and r the number of rows, and the arrow indicates the interdomain interface.

The average frontwidth, F, is expressed as:

$$F = 7(2Sa(1) + Sa(2)) + 7r + 2 \quad , \tag{7.1}$$

and the number of variables V as:

$$V = (10Sa(1) + 10Sa(2) - 13)r + 4 \quad , \tag{7.2}$$

so the work load, Wa, in the sector is:

$$Wa = [7(2Sa(1) + Sa(2)) + 7r + 2][\{10(Sa(1) + Sa(2)) - 13\}r + 4] \tag{7.3}$$

Case B:

This sector defined by two segments and no interdomain interface.

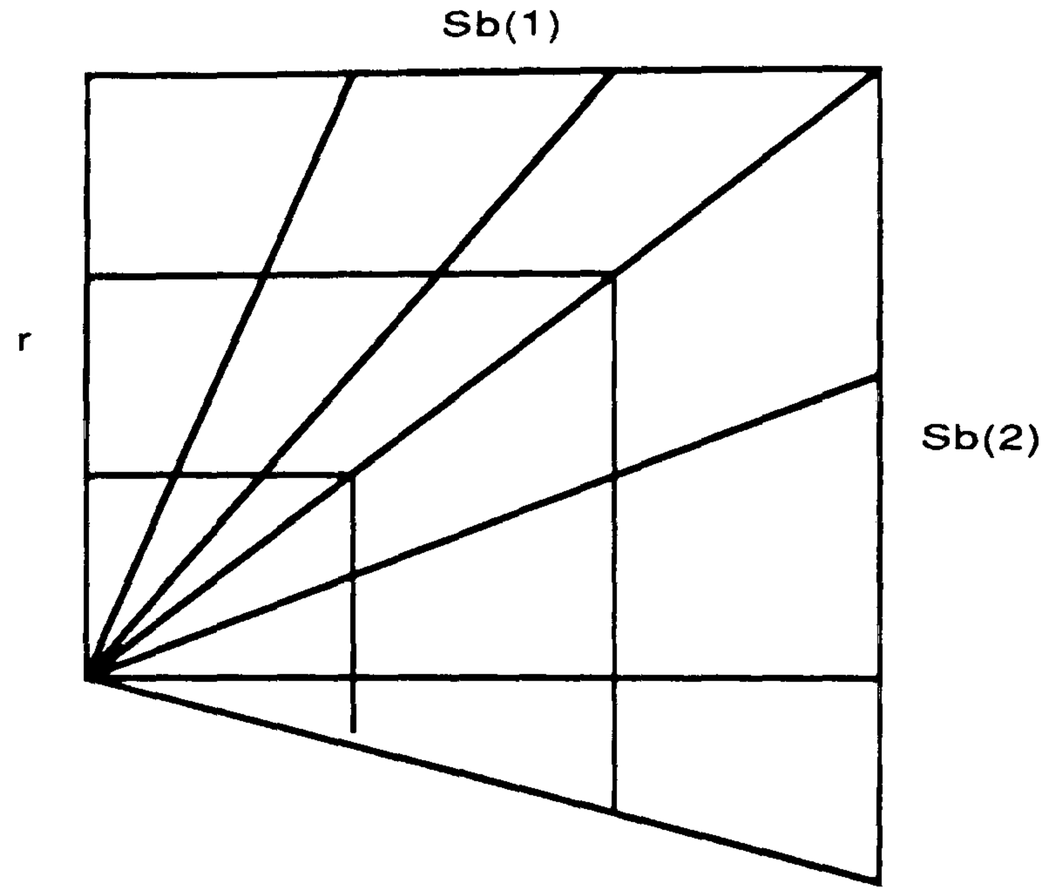


Figure 7.12

Here Sb(1) and Sb(2) signify the number of spokes in their respective segments and r the number of rows. In this case the average frontwidth, number of variables and work load are:

$$F = 7(Sb(1) + Sb(2)) + 7r + 9 \quad , \tag{7.4}$$

$$V = (10\ Sb(1) + 10Sb(2) - 13)r + 4 \quad , \tag{7.5}$$

$$Wb = [7(Sb(1) + Sb(2)) + 7r + 9][\{10(Sb(1) + Sb(2)) - 13\}r + 4] \quad . \tag{7.6}$$

Case C:

This sector defined by one segment and one interdomain interface.

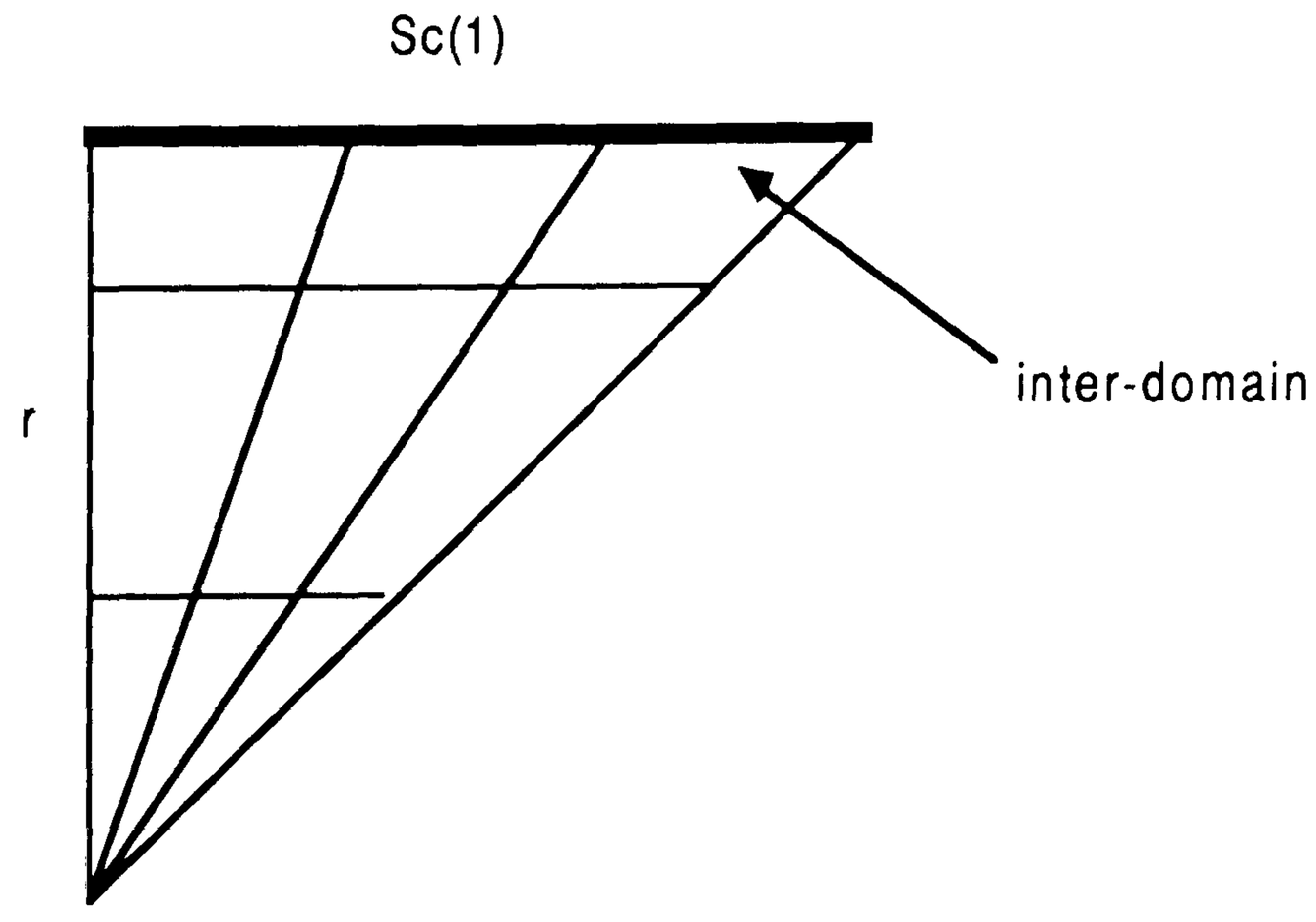


Figure 7.13

Here Sc(1) signifies the number of spokes in the segment and r the number of rows, and the arrow indicates the interdomain interface. Here

$$F = 14Sc(1) + 7r + 2 \quad , \tag{7.7}$$

$$V = (10Sc(1) - 3)r + 4 \quad , \tag{7.8}$$

$$\text{and} \quad Wc = [14Sc(1) + 7r + 2][(10Sc(1) - 3)r + 4] \quad . \tag{7.9}$$

Case D:

This sector defined by one segment and no interdomain interface.

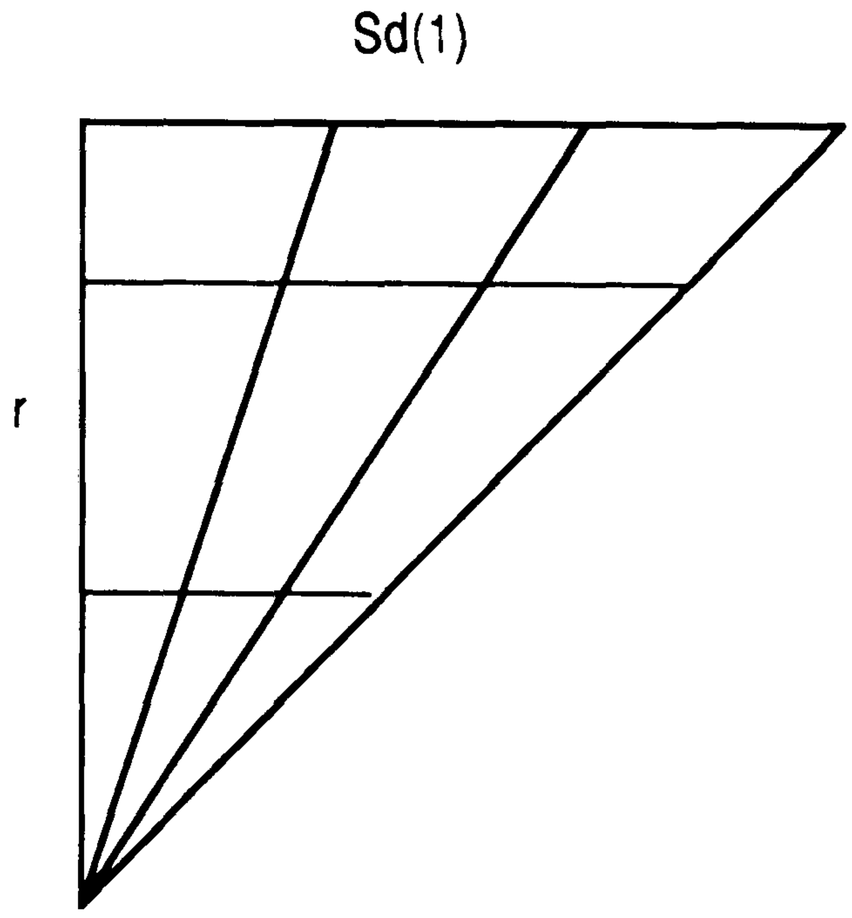


Figure 7.14

Here Sd(1) signifies the number of spokes in the segment and r the number of rows. And so

$$F = 7Sd(1) + 7r + 16 \quad , \tag{7.10}$$

$$V = (10Sd(1) - 3)r + 4 \quad , \tag{7.11}$$

$$Wd = [7Sd(1) + 7r + 16][(10Sd(1) - 3)r + 4] \quad . \tag{7.12}$$

A sector of type A is classified as the most constrained of the four cases. Two reasons are given for this choice:

(i) The number of spokes in a sector has to be a positive integer. Although in calculating the number of spokes in each sector, from applying the above equations, the value would not normally be an exact whole number and so in rounding the value an error would be introduced. This error would propagate throughout the calculation of the

work loads in each sector, causing them to be out of balance, the magnification of the error being greatest for the interdomain type sector. The reason for this arises from the coefficients of the number of spokes terms in the formula for the frontwidth being larger for the interdomain type sector. As we are required to specify the number of spokes for one of the sector types it is obvious that an interdomain type sector should be chosen so that no error would be incurred by this sector, as the errors incurred by the other sectors are comparatively smaller.

(ii) For a sector of type A we have one equation and two unknowns. In equation (7.3) it is impossible to combine Sa(1) and Sa(2) to make one unknown because the expression in the first bracket (2Sa(1)+Sa(2)) does not equal the expression in the second bracket (Sa(1)+Sa(2)) and so we are unable to solve the equation. Therefore it essential that if a sector of type A is present in the domain it should be specified as the most constrained sector.

If a sector of type A is not present in the domain then a sector of type C should be chosen as it also satisfies condition (i) above.

.

From the way in which the domain has been subdivided each sector is checked to see which category it belongs to. If there are type A sectors present, which are the most constrained sector type, the user defines the number of spokes and rows required to cover this sub-region. For the case when type A is not present, type C is the next most constrained and would require the spokes and rows to be specified.

From the information specified for the most constrained sector a value for the work load Wa (or Wc) is calculated from the simple expression (7.3) (or (7.9)). The equations for the other cases can now be equated to the value obtained (Wa). To balance the work load over the remaining sector types it is necessary to solve a quadratic, the final equations for cases B,C and D are given below:

Case B:

With $Wa = Wb$ and $B = Sb(1) + Sb(2)$,

$$Wa = 70rB^2 + (70r^2 - r + 28)B - (91r^2 + 89r - 36) \quad . \tag{7.13}$$

The quadratic is solved in the standard way for B and only the +ve value of the square root is considered. The value of B is rounded to give an integer value for the number of spokes. This number is then divided to give Sb(1) and Sb(2).

Case C:

With $Wa = Wc$ and $C = Sc(1)$,

$$Wa = 140rC^2 + (70r^2 - 22r + 56)C - (21r^2 - 22r - 8) \quad . \tag{7.14}$$

Once again the quadratic is solved and Sc(1) is rounded to give an integer value.

Case D:

With $Wa = Wd$ and $D = Sd(1)$,

$$Wa = 70rD^2 + (70r^2 + 139r + 28)D - (21r^2 + 20r - 64) \quad . \tag{7.15}$$

Again Sd(1) is calculated from the quadratic and rounded to give an integer number. There will be a degree of error incurred due to rounding the value, as it is unlikely that the value returned will be a whole number. The efficiency of the load balancing strategy will be examined in section 7.5.

7.3.2 Sector Ordering

The importance of sector ordering is to maintain an optimal front size at each stage. The organisational binary tree, fig7.1, is used to decide on which processors each sector are best placed. We require at each level of the tree that adjoining sectors in the domain should be placed on adjacent processors whose generated information is to be coalesced. It is not sufficient to only satisfy these conditions at level 0 of the tree, it is also necessary that higher levels of the tree be satisfied; i.e. neighbouring enlarged sectors should also be placed on adjacent processors whose results are to combined. This is best explained by an

example; consider the binary tree for eight processors, fig.7.1, and the disk rotor test problem with a typical partitioning of the domain for eight processors shown in fig. 7.8.

At level 0 the processors work independently and no coalescing takes place. The first instance of combining sectors occurs at level 1 where the sector on processor 1 is coalesced with the sector on processor 0, similarly the the sectors on the other processors are combined as (2,3), (4,5) and (6,7). For each bracketed pair both sectors should be adjacent in the domain as shown in fig. 7.8. At level 2 of the tree, the results from processor 0 are coalesced with those from processor 2 and consequently the enlarged sector on processor 0 should be neighbouring the enlarged sector on processor 2, likewise the enlarged sectors on processors 4 and 6 are combined. This is shown diagrammatically in fig.7.15.

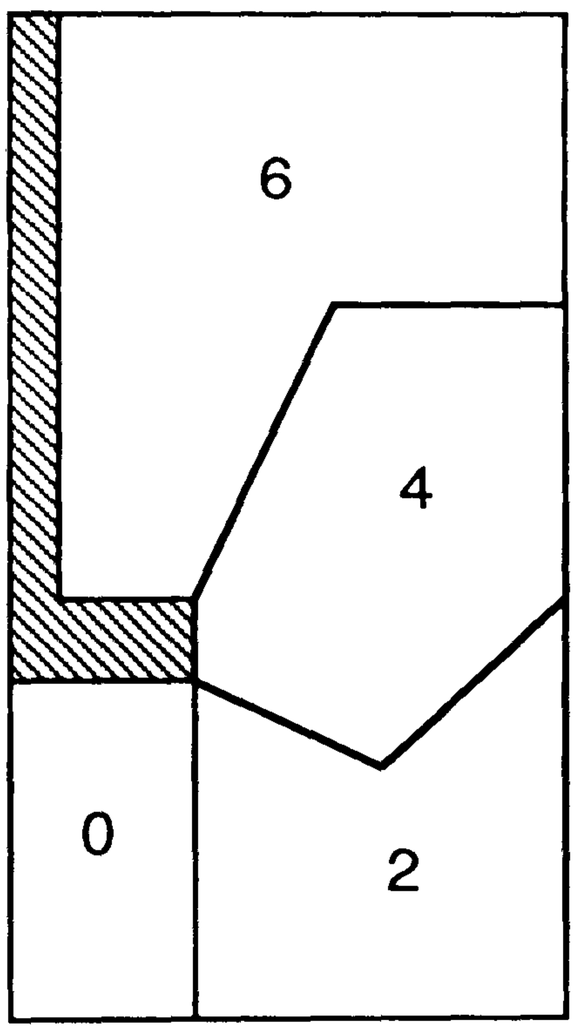


Figure 7.15

On reaching the top level of the tree the conditions are naturally satisfied as the final two enlarged sectors to be coalesced represent the domain as a whole, processors 0 and 4, shown in fig.7.16.

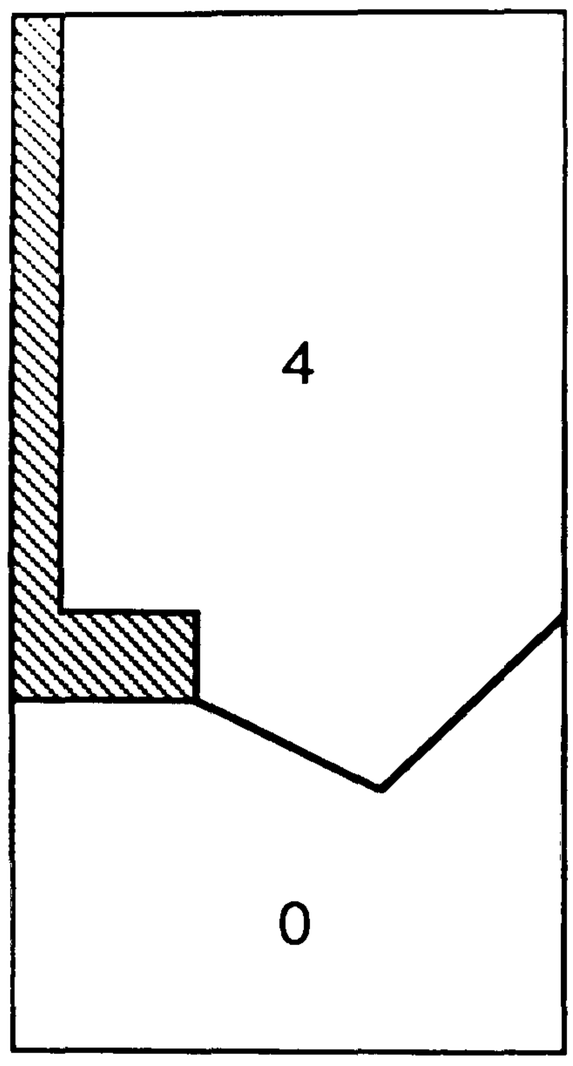


Figure 7.16

7.3.3 **Mesh Refinement**

To aid the explanation of mesh refinement a very coarse mesh (fig7.17) is presented for the disk geometry which will be modified to show the different types of refinement that are possible. In designing a mesh for a non-convex domain it is necessary to apply the load balancing strategy described earlier to guarantee an even loading across each processor. The meshes that are displayed in this section are simply meant to show the many methods of refinement that can be employed and do not represent load balanced meshes.

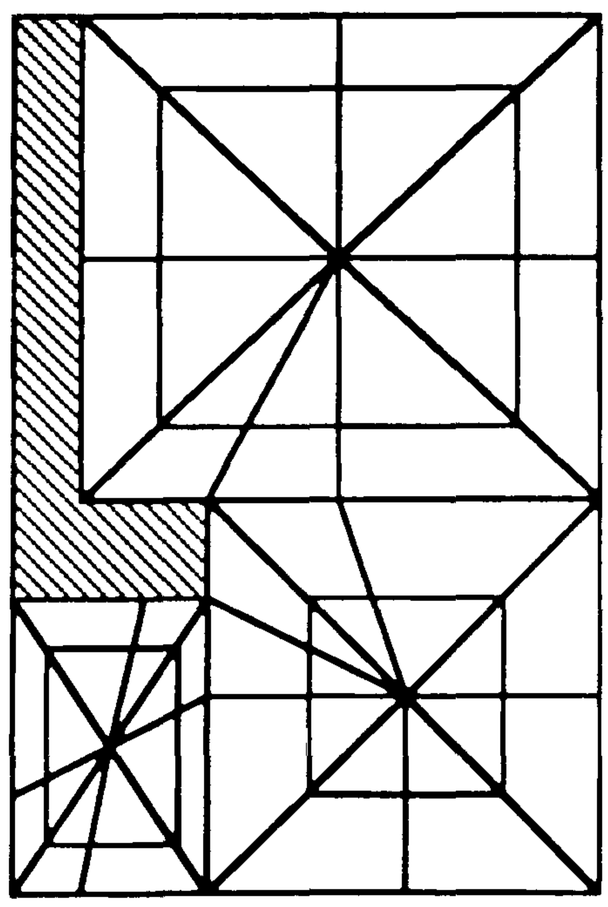


figure 7.17

Mesh refinement can be achieved in a similar way to that specified for the convex region. For a grid which is generally too coarse it can be refined overall by increasing the number of spokes and rows on the most constrained sector and then applying the balancing strategy to specify the meshes covering the other sectors. An example of overall refinement is shown in fig.7.18.

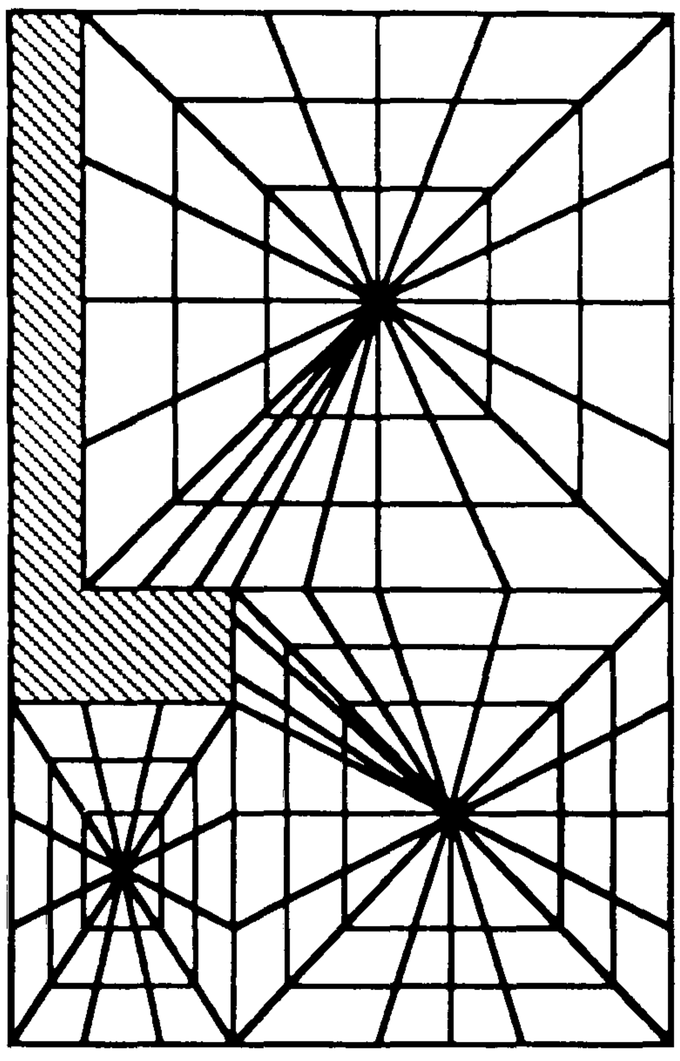


Figure 7.18

Refinement close to the boundary is attained by having the rows close together near the boundary and further apart elsewhere. The spacing between spokes can also be varied. An example of this type of refinement is shown in fig.7.19.

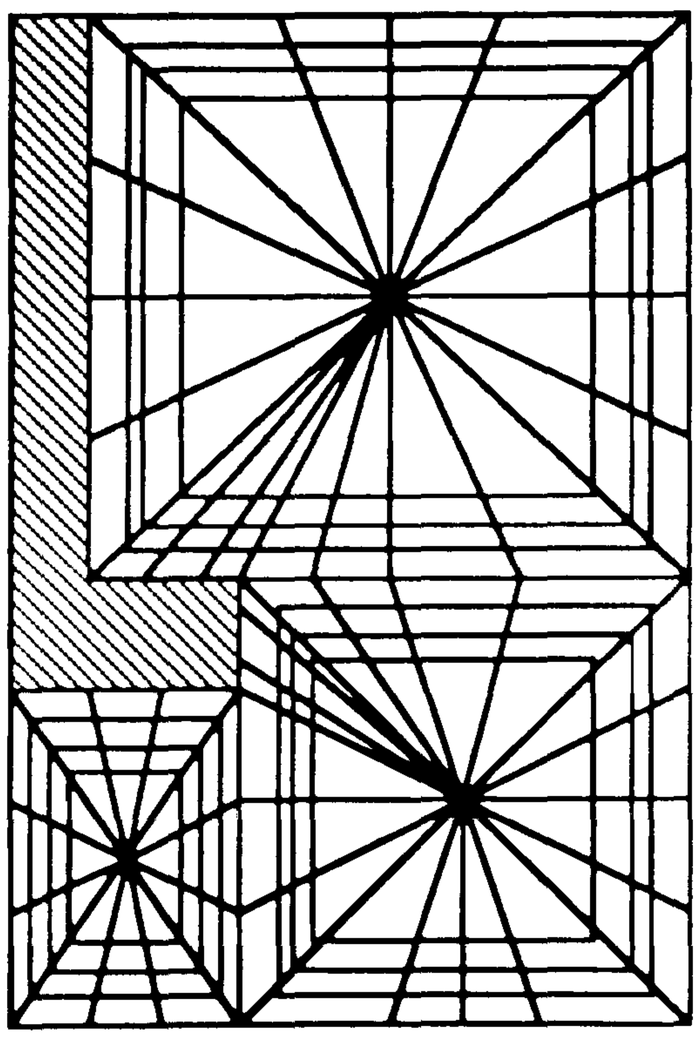


Figure 7.19

For refinement near the rotor the sub-domain centres can be moved closer to the actual rotor causing the elements to be condensed in this area. An example of this type of refinement is shown in fig.7.20.

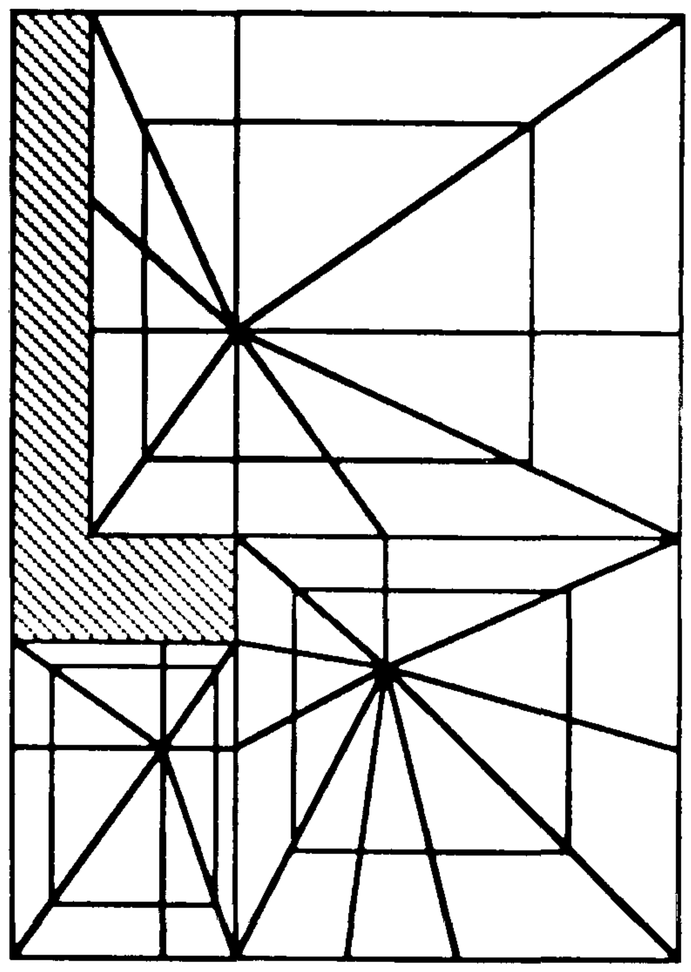


Figure 7.20

Similar methods of sector ordering and refinement can also be employed for the cylinder rotor problem.

7.4 Program Description

From the network shown in the previous chapter fig.(6.6) a root processor is connected to a network of eight transputers in a bi-directional ring configuration. Similarly four and sixteen processor configurations have also been used. Also attached to the root processor is a graphics board and screen. The program is composed of two major parts, (a) the control process and (b) the multifrontal solver process. Consideration will be given to each part seperately, giving a brief outline of how the program works. The program has been completely written in Occam 2.

7.4.1 The control process

Placed on the root processor is a pre-processing algorithm which subdivides the domain between the number of processors in the network and generates mesh information for each individual processor as a separate entity. This information includes the (x,y) coordinates for each node, element topology, boundary conditions and interface nodes for the specific subdivision. Added to this information is a global mapping which defines the local mesh ordering to the domain as a whole. The information is tagged by the number of its destination processor and is then passed via channels from the root processor to its destination on a particular processor. This personalised information is sent in packets which include a specific protocol composed of a header stating its destination processor and original sending processor , also the sizes of the arrays used followed by the arrays themselves. A specific routing algorithm is used which is dependent on the link configuration. The routing algorithm directs the packets through efficient paths around the network, via specific links and processors, until they reach the specified processor.

The partial differential equations used are non-linear in the advection term and a simple iterative scheme is employed as the method of solution, since our main aim initially is to implement the algorithm and test its feasibility as a good parallel algorithm. At a later stage a Newton procedure may be adopted to speed up convergence. The root processor maintains overall control of the solution procedure. On completion of each iteration each processor in the network sends information to the root stating the maximum velocity difference between successive iterates in its sub-region. From this information the root performs a check on whether the solution has converged to a predefined tolerance. On failure to reach this set limit a message is sent to all processors in the network to perform another iteration.

When the convergence criteria has been met the root sends another message to the network, this time to calculate the stream values and also the colour values for predefined contour levels. The colour values are sent back from the network to the root

which sends the correct commands to the graphics board for the contour plot to be displayed on the screen, sector by sector. For the program to complete successfully finish instructions are sent to each processor in the network.

7.4.2 The multifrontal solver process

An identical copy of the multifrontal program segment is placed on each processor in the network except for processor 0. Processor 0 has some extra tasks to perform as it is the link between the network and the root and requires additional code to send/receive information to/from the root. For this reason a special version of the harness has to be loaded onto processor 0. Processor 0 receives the initial personalised mesh data for each processor and sends it via the routing algorithm to the correct destination processor. Each processor receives its information and proceeds to work on its own individual data. Initially it sets up a number of constants namely the size of initial frontwidth, initial or previous values of the velocities and also an occurrence array is defined for the number of appearances of each node before it is fully summed.

The first stage of the multifrontal procedure begins by assembling the first element matrix. The element matrix is described in chapters 4 and 5 for the primitive u,p,v and u,p,v,w approaches; the elements are either eight noded quadrilaterals or six noded triangles. The procedures used in the fluid element matrix routine are similar to those found in the NAg finite element library [34], and our Occam routines were verified by comparing with results produced by equivalent NAg Fortran routines on the mainframe Vax super-mini computer. The necessary modifications must also be made to the occurrence matrix each time an element is assembled.

On entering the assembly and elimination stage, level 0 of the tree, a loop counter is specified for the number of variables to be eliminated and elements to be assembled. New element matrices are assembled provided that a number of conditions are satisfied, i.e. the front width is less than its critical value, that there are more elements to be assembled

and that none of the fully summed variables are boundary variables. If these conditions are not satisfied a pivot is found, its row and header stored and the row eliminated from the frontal matrix.

On completion of the first stage, level 0 of the tree, there will be a number of variables left in the matrix and these will be the variables which lie on the sector interfaces. From the number of processors present (2^m) it is straightforward to determine the number of levels in the tree (m+1); i.e. for eight processors there are four levels (0 to 3). For the disk rotor test problem a typical partitioning of the domain for 8 processors is shown in fig.7.8.

The next stage to be performed is that of level 1 where, depending on the processor number, the results from level 0 are either sent or received by the processors shown in the tree. This information is again sent in packets using the routing algorithm to direct it to its destination. The processor numbers begin from zero, and the even numbered processors receive the results from their odd numbered counterparts. The received frontal matrix is then coalesced into the frontal matrix already residing on the particular processor and again modifications to the occurrence array takes place.

Following this a number of fully summed variables, those lying on adjoining sector interfaces, are chosen in order of magnitude of their pivots. Their rows and headers are saved and they are successively eliminated from the matrix. Modifications to this stage have been made where the work is distributed between processors which are idle adopting a master slave approach, as explained in section 7.1. Figure 7.15 shows the domain after the first coalescing has occurred. The same procedure is preformed for the remaining levels and Fig. 7.16 shows a diagram of how the sectors are combined at the next level.

When the final level is reached a specific number of variables are left, the actual number remaining depends on the domain type. For the convex region only variables defined at the domain centre remain, whereas for the non-convex region the remaining variables are

those defined at the sub-domain centres plus the sub-domain interfaces. Now all of these variables are in a suitable state to be eliminated.

It is worth noting for the non-convex region that some of variables at the sub-domain centres and sub-domain interfaces may be fully summed at a level lower than the top level and therefore are eligible for elimination. Even though such instances can occur it is preferred not to eliminate them at this juncture, the reason for this being that it would cause difficulty in keeping track of their positions and consequently they are only eliminated at the top level. They are also eliminated in a specific order, firstly the sub-domain interfaces and then the sub-domain centres.

Following this the back substitution is performed, which means beginning at level 3, the root, and traversing the tree in reverse order down to the leaves, level 0. This is achieved by calculating values for the variables at a particular level and then passing the values and variables to the specific processor indicated by the tree. This process is repeated until the bottom level is reached, where each processor then calculates the values for the interior variables of its sector. During this part of the algorithm some processors are idle waiting for values to be passed to them from higher levels, however at the lowest level all are active working on their own sector. When this stage has been accomplished a quick check is made to find the largest velocity difference between iterates in the particular sector and this information is then sent to the root processor.

When the solution for the velocities has converged as indicated by the root processor, a message is sent to the network to calculate the stream values. This again is performed in parallel, the same multifrontal program is used except this time a different element matrix procedure is called. The element matrix for the stream function is presented in chapter 4 for the 2-D case and chapter 5 for the 3-D case, the equations are linear and are therefore solved directly. From the stream values obtained the associated colour values are calculated.

The colour values are also evaluated in parallel, each processor in the network working on its own sector. Each element in a sector is considered separately and colour values are determined at each screen pixel inside the element for pre-defined contour levels. These global elements are mapped onto a local general element where the shape function routine is used to calculate the stream function values at each pixel point inside the element. From this knowledge the colour values at the required contours are found.

Some care should be taken when mapping the elements as a knowledge of which sides of the global element map to the particular sides of the local element is essential, otherwise the contours may be drawn in the wrong direction inside the element. Consider a triangular element (i) ,see fig.(7.21), which is mapped onto a local triangular element (ii). The side a,b of (i) is mapped onto the side 1,2 of (ii) and the stream function values are evaluated for each pixel a row at a time.

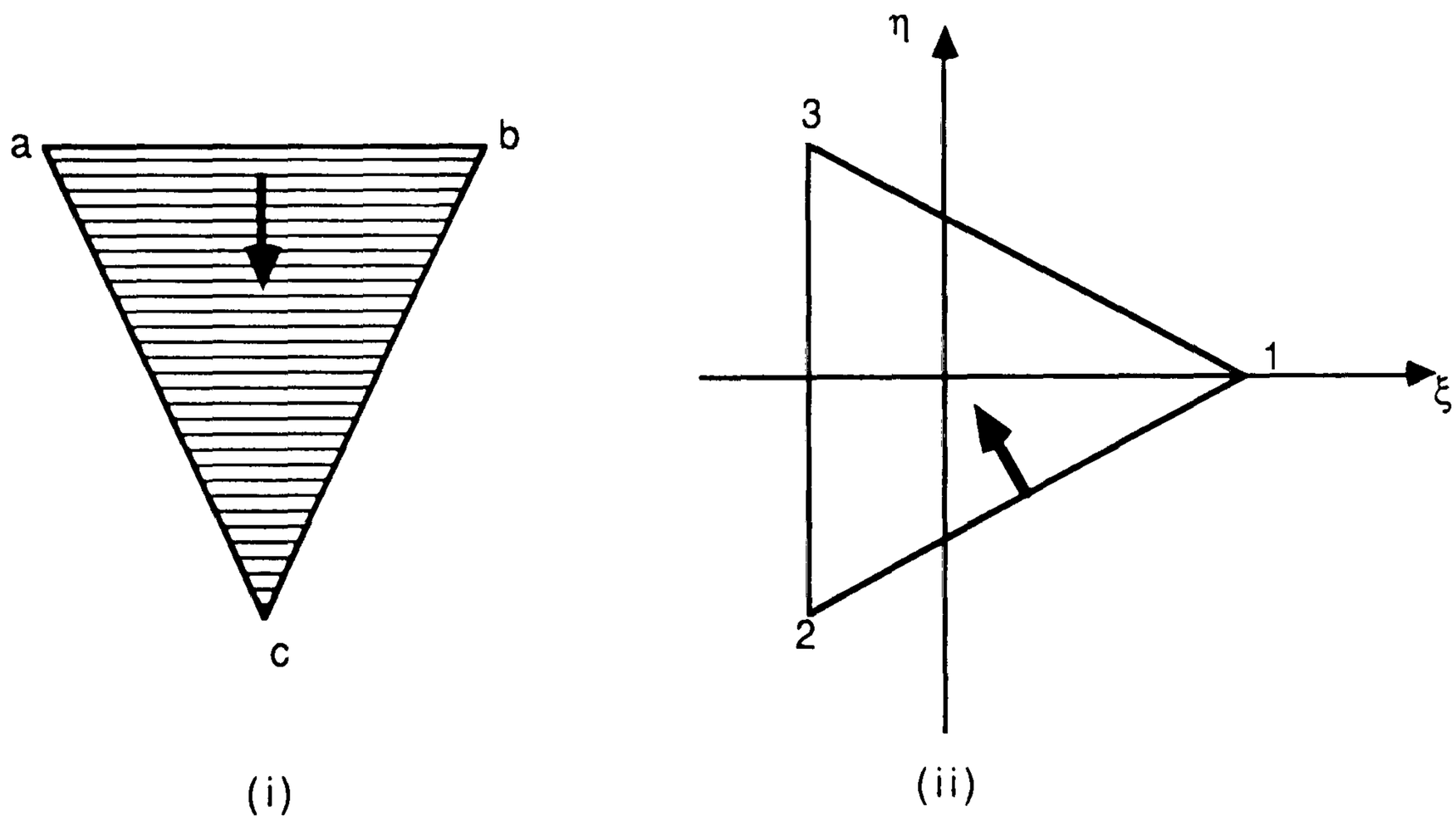


Figure 7.21

Beginning at the line 1,2 and progressing in the direction of node 3 we calculate values for each row of pixels until they converge on node 3. We then map back to the global element remembering that the first row of values is defined along a,b and subsequent rows are parallel to a,b progressing towards c.

When all of the colour values in a sector have been calculated, the colour values are compressed along the rows, noting only when a change in the colour value has occurred.

Then information concerning the plotting is sent back to the root processor.

The graphics routines that have been developed are very simple yet produce an accurate graphical representation of the flow being modelled. It was not our intention to implement a sophisticated set of graphics routines but to produce a visual representation that would enable us to view the results at a glance.

7.5 **Results**

In this section results are presented for the two types of fluid flow problem mentioned earlier; 2-D driven flow over a cavity (convex domain) and 3-D axisymmetric flows in mixing (non-convex domain). The results were obtained using the aforementioned concurrent algorithm on networks of 4, 8 and 16 transputers.

7.5.1 **Driven flow over a square cavity at Re=100.**

The finite element mesh employed has a mixture of six noded triangular elements and eight noded quadrilateral elements with 15 and 20 degrees of freedom per element respectively. In order to evaluate the efficiency of the algorithm it is convenient to split the algorithm into three stages and to perform timings over these stages;

1) the level 0 eliminations of fig. 7.2,

2) the other levels of the elimination phase in fig. 7.2,

3) the back-substitution.

At stage (1) the transputers are working independently whilst in stages (2) and (3) some communication between processors is also taking place. The time spent in the three stages were recorded for a single iteration of the scheme. The results presented graphically in fig.7.22 for 551, 1099 and 2195 variables on networks of 4, 8 and 16 processors respectively.

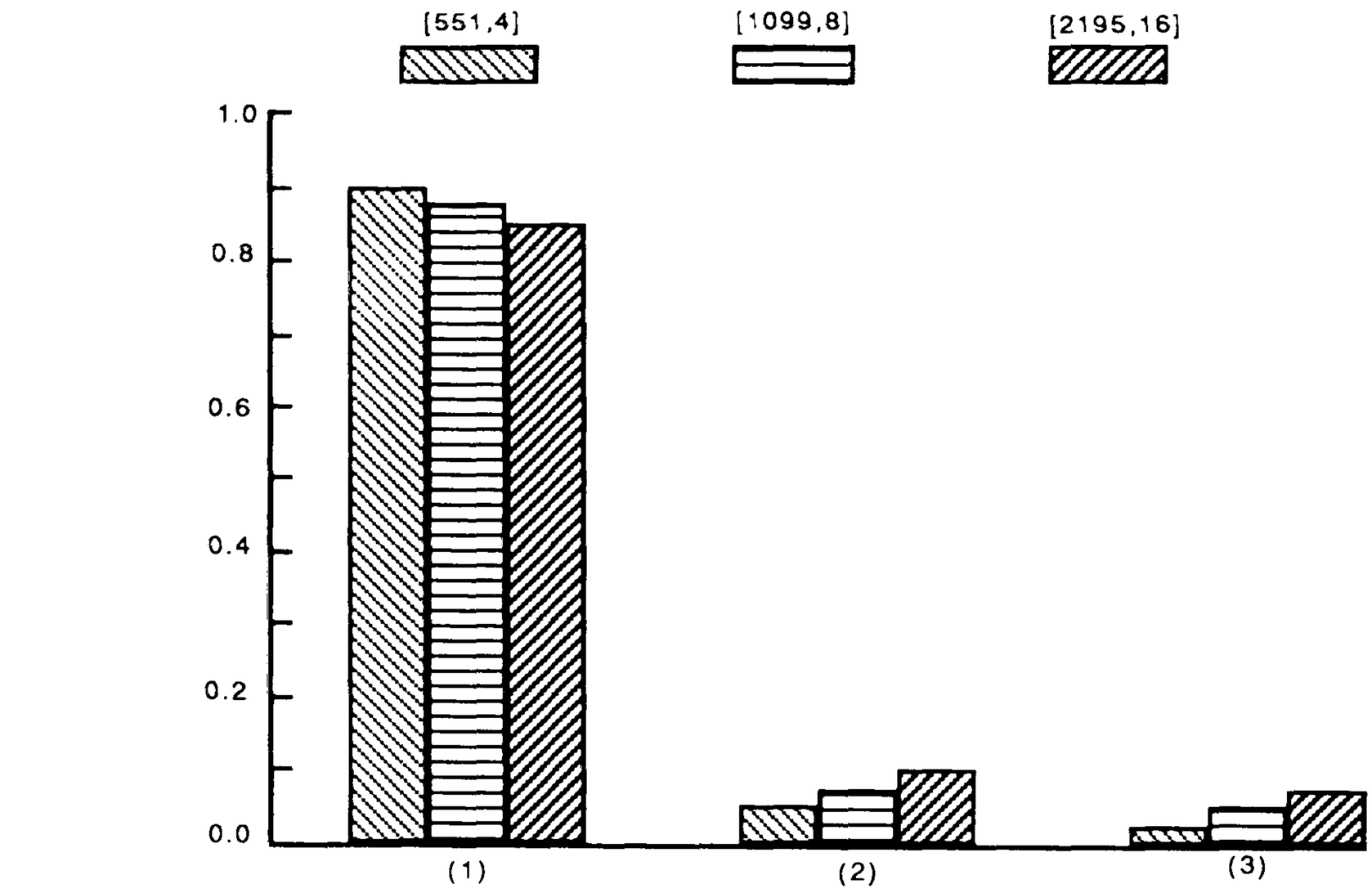
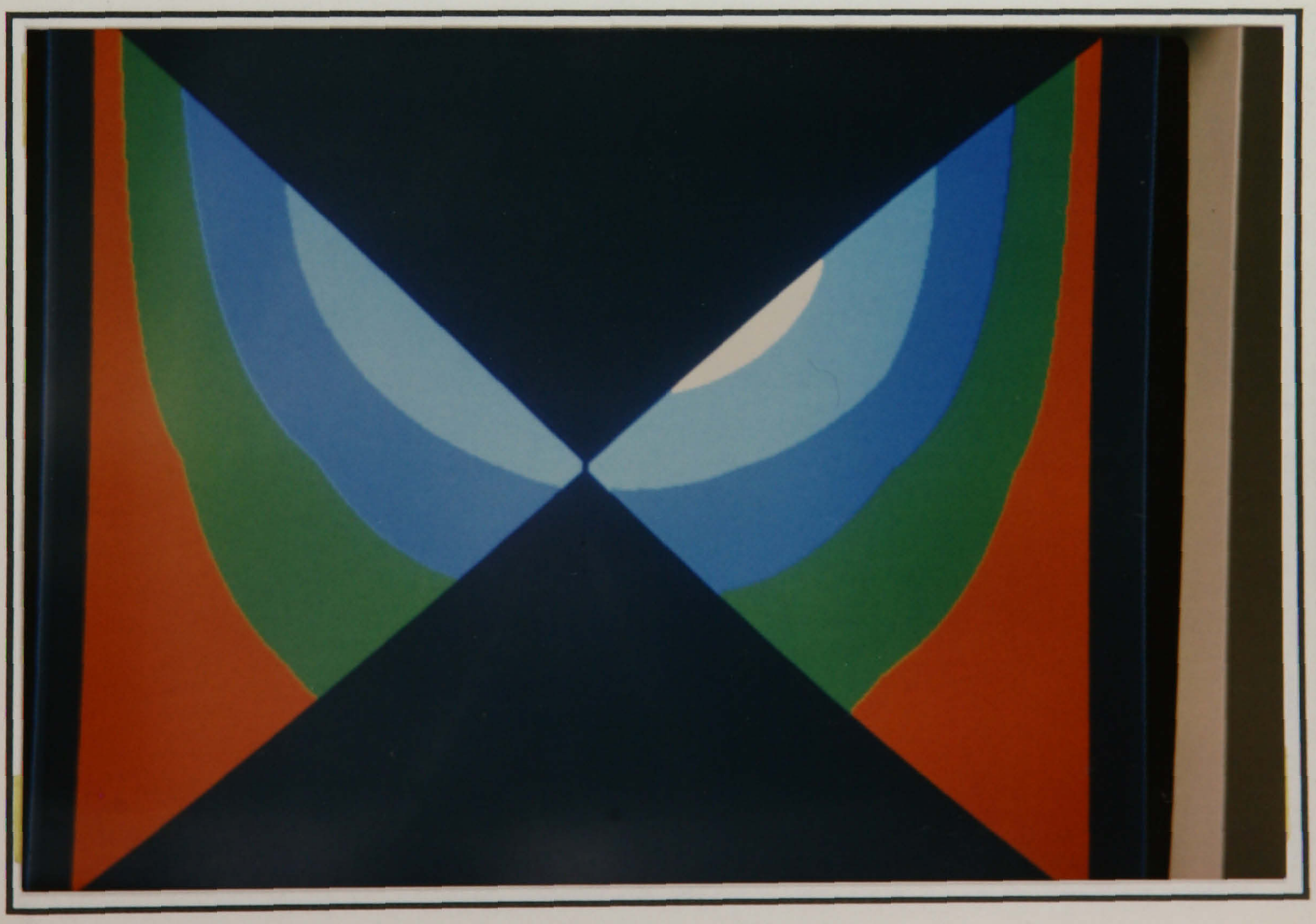


Figure 7.22

Here the pair [551,4] signifies 551 variables and 4 processors. From the results presented it is estimated that a processor efficiency exceeding 85% is achieved. The amount of time taken to complete one iteration for the above sets of test data are as follows; the [551,4] set takes 9.3 seconds, the [1099,8] set takes 10.5 seconds and the [2195,16] set takes 11.8 seconds. These results indicate that it is possible to attain almost linear speed-up; a slight degradation in the times is incurred as the number of processors in the network is increased. When the program is allowed to run to completion the streamfunction values are obtained and a plot of the streamlines is presented on the screen, which are in excellent agreement with Cliffe et al [15] and also with our earlier results produced on the Vax, see chapter 4.

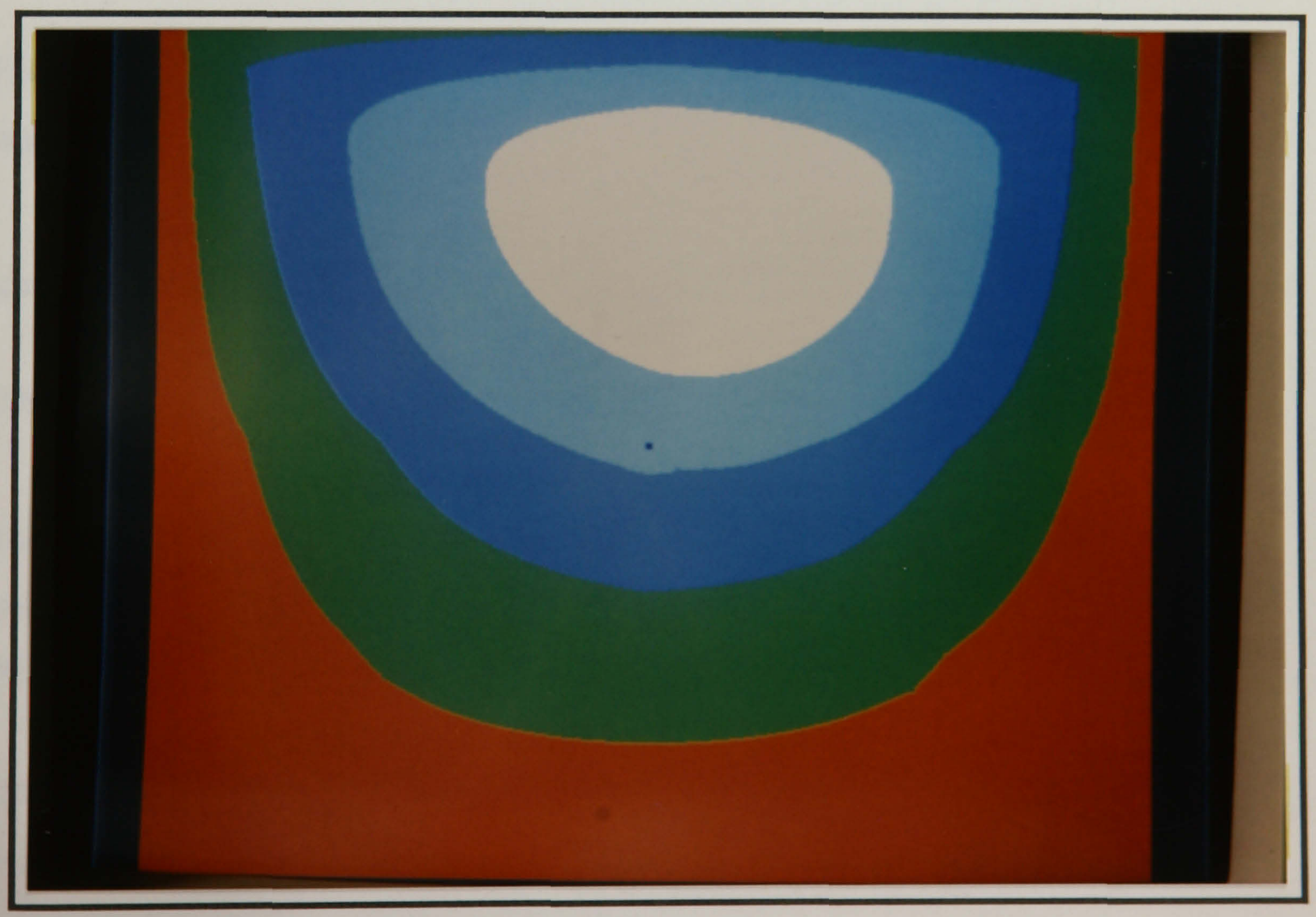
Presented below are four photographs of the graphics screen displaying the streamlines for the cavity problem at different stages of its generation. The photographs show a picture of how the flow domain is built up sector by sector as the graphical information is sent back by each processor.

Photograph 7.23



Photograph 7.24

Photograph 7.25



Photograph 7.26

7.5.2 **Axisymmetric mixing at Re=25.**

In this section results are discussed for both the disk and cylinder rotor geometries. The finite element mesh is composed of both triangles having 21 degrees of freedom and quadrilaterals having 28 degrees of freedom. The load balancing strategy described in section 7.3.1 was employed.

7.5.2.1 **Disk rotor geometry**

The efficiency of the load balancing strategy is examined first of all. The partitioning of the domain is shown in fig.7.8, where sectors 1,3,4,5 are of type A and sectors 0,2,6,7 are of type B. We have

number of rows = 5 ;

number of spokes for Sa(1) = 6 and for Sa(2) = 3 ;

interdomain interface lies on Sa(1).

Firstly we calculate the work load on A (Wa) (most constrained sector) from (7.3) as

Wa = 142*389 = 55238 .

Using (7.13) we calculate Sb(1) and Sb(2) as

$$(Sb(1) + Sb(2)) = \frac{-1773 + 9177.9262}{700} = 10.578 \ ,$$

Sb(1) + Sb(2) = 11 (rounded) ,

Sb(1) = 5 and Sb(2) = 6 ,

Wb = 121*489 = 59169 .

Experimental timings after completing level 0 eliminations are:

First transputer back: 155.569 secs ,

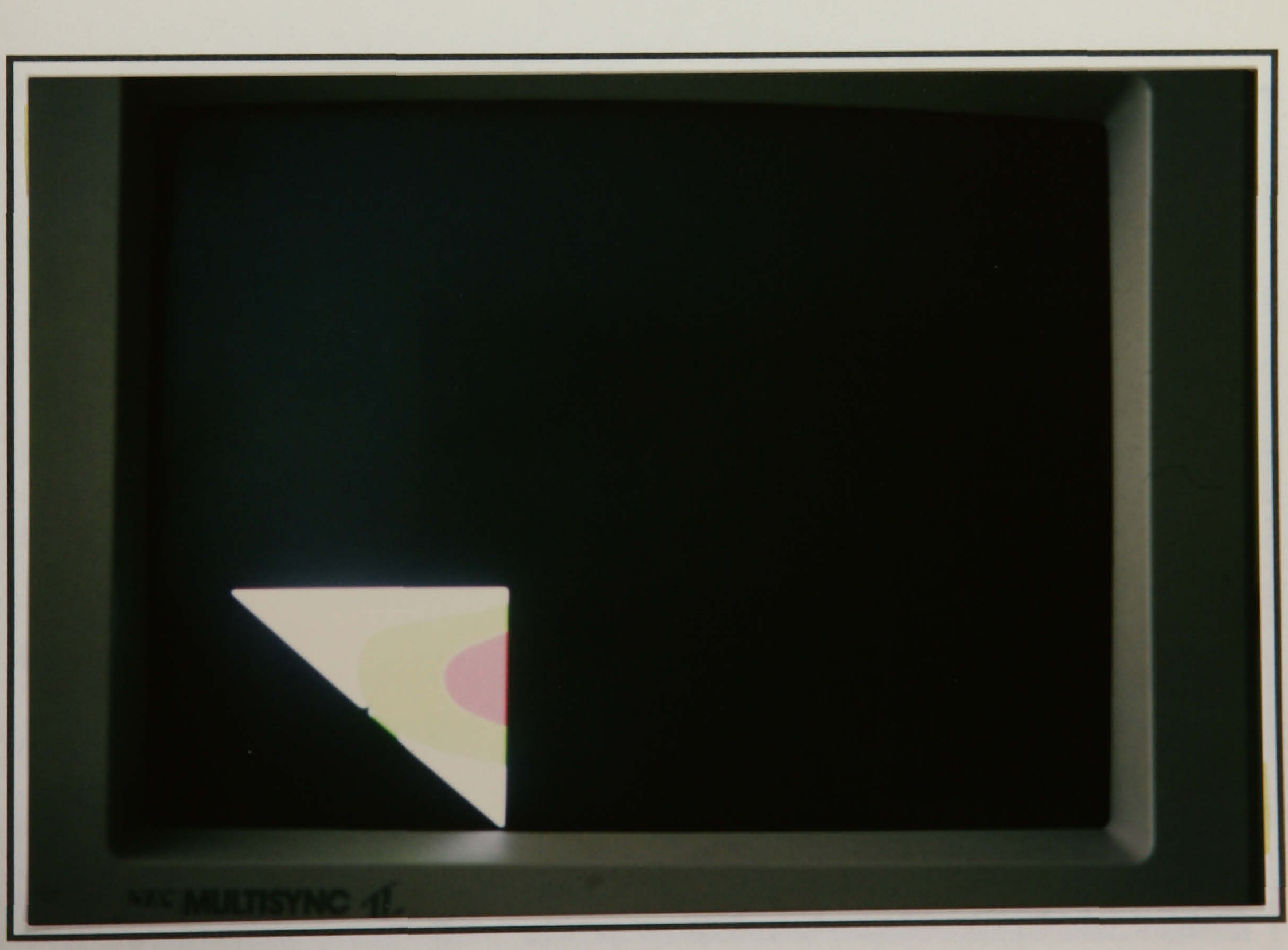
Last transputer back: 162.311 secs ,

with percentage difference: 4.3% .

This signifies that the load balancing strategy produces good results with only a 4.3% discrepancy recorded.

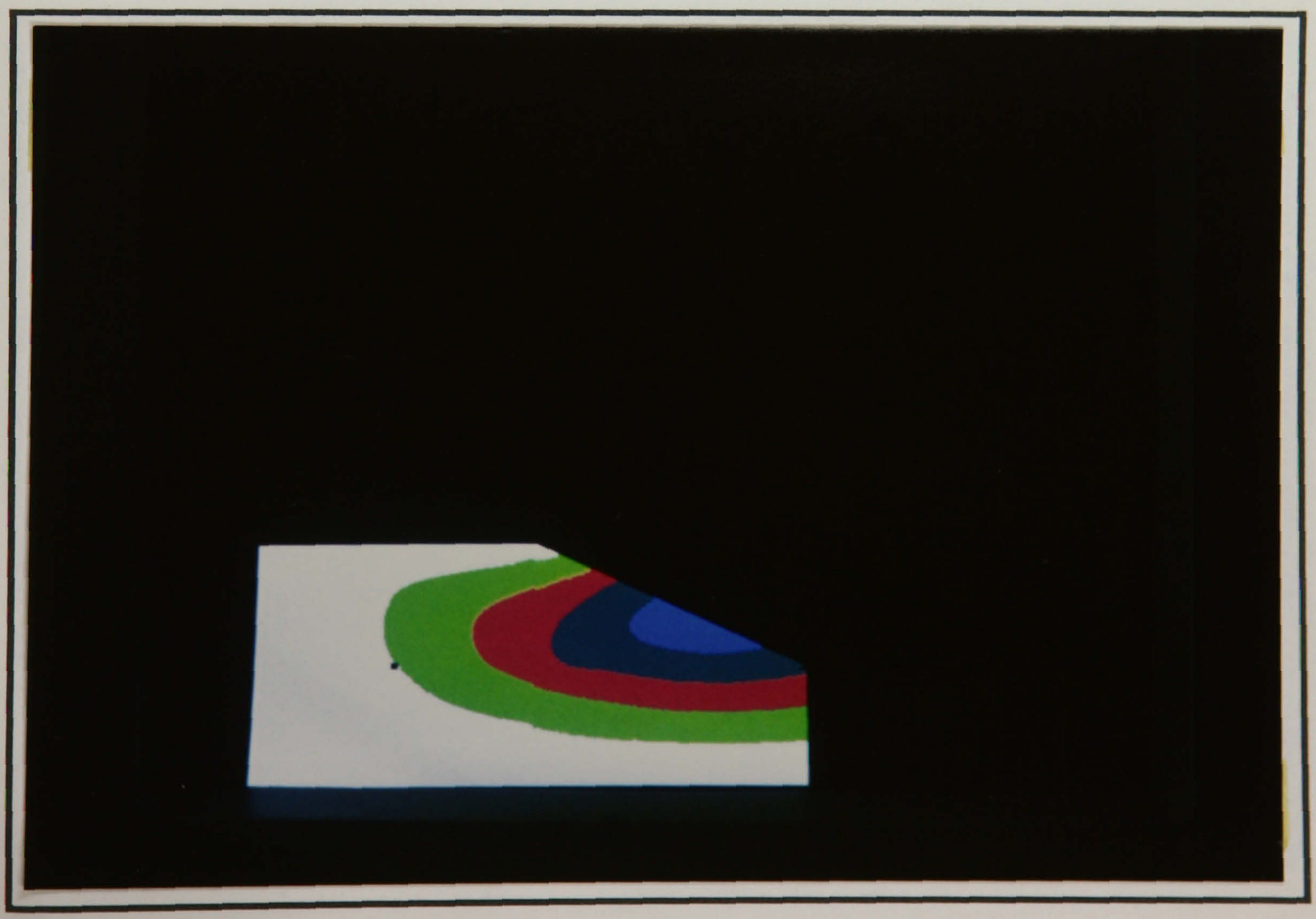
This disk problem has been solved using 4391 variables on a network of eight transputers, taking 121 seconds to complete a single iteration with a 4% discrepancy in the load balancing. For this type of problem the processor efficiency achieved exceeds 85%.

Presented below are eight photographs of the graphics screen displaying for the eight separate sectors sections of the streamlines showing how the picture is built up from the information sent back by each processor. The results obtained are in good agreement with the work presented in chapter 5 produced on the Vax and also with the that of Spragg [103].

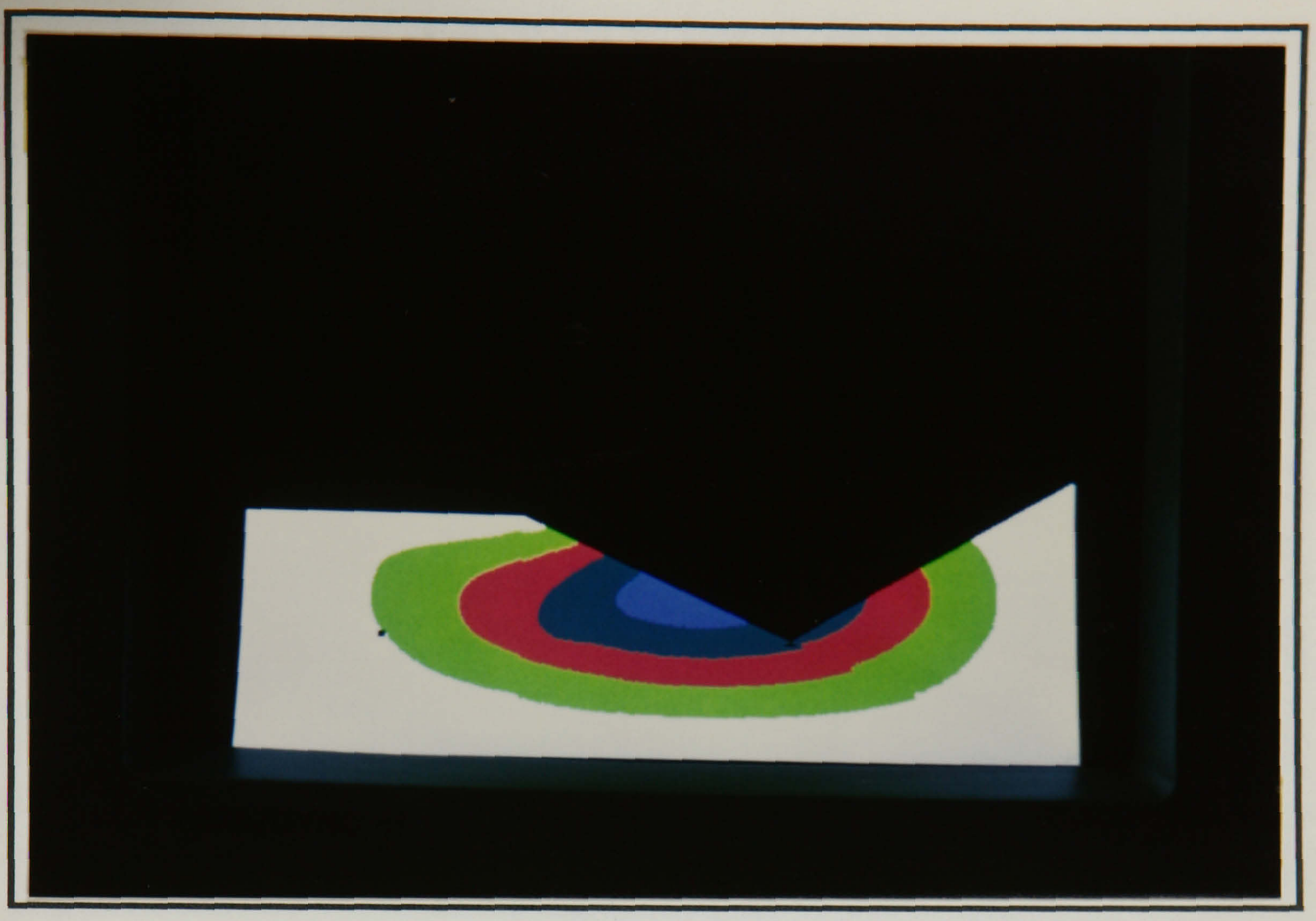


Photograph 7.27

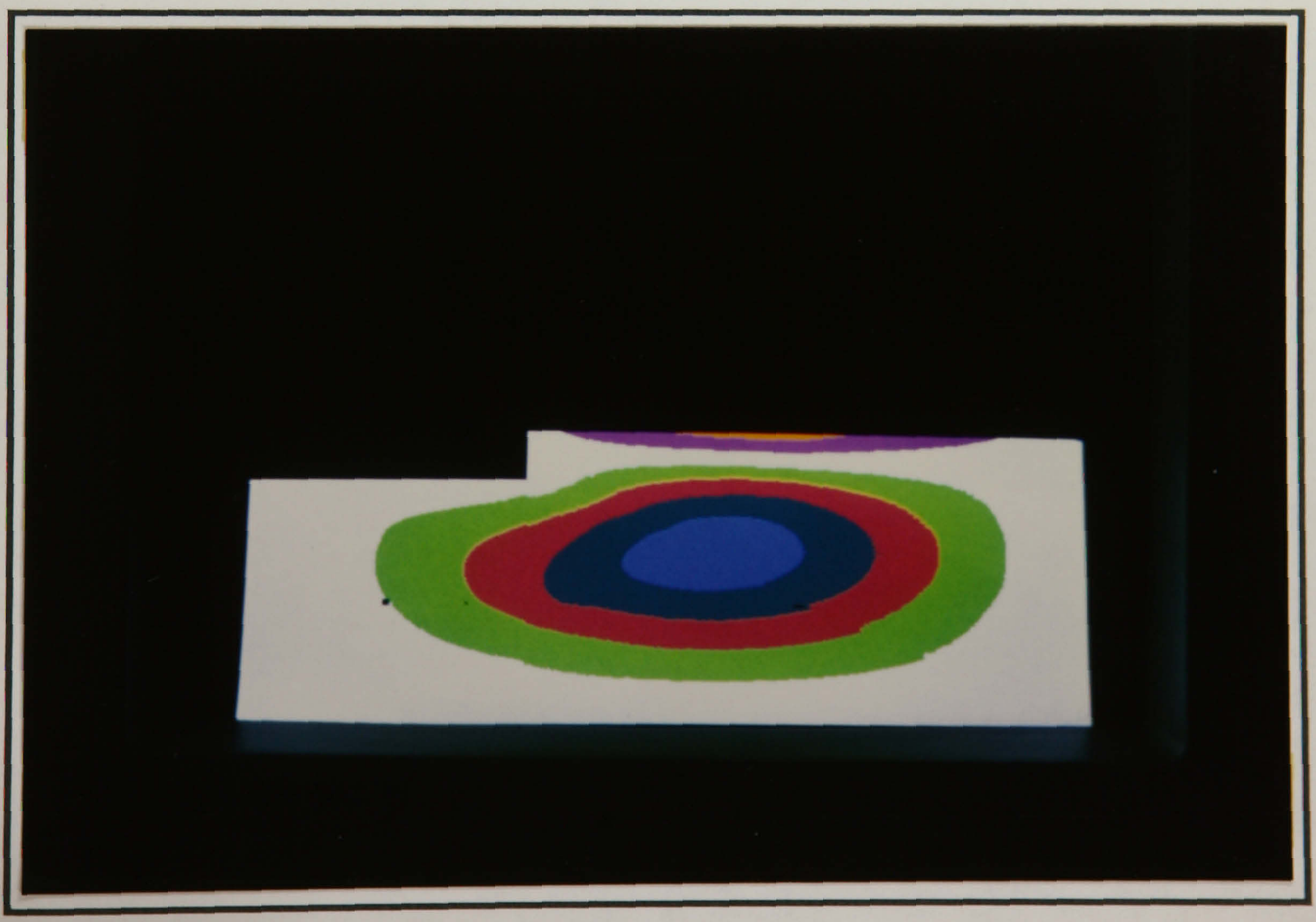
Photograph 7.28



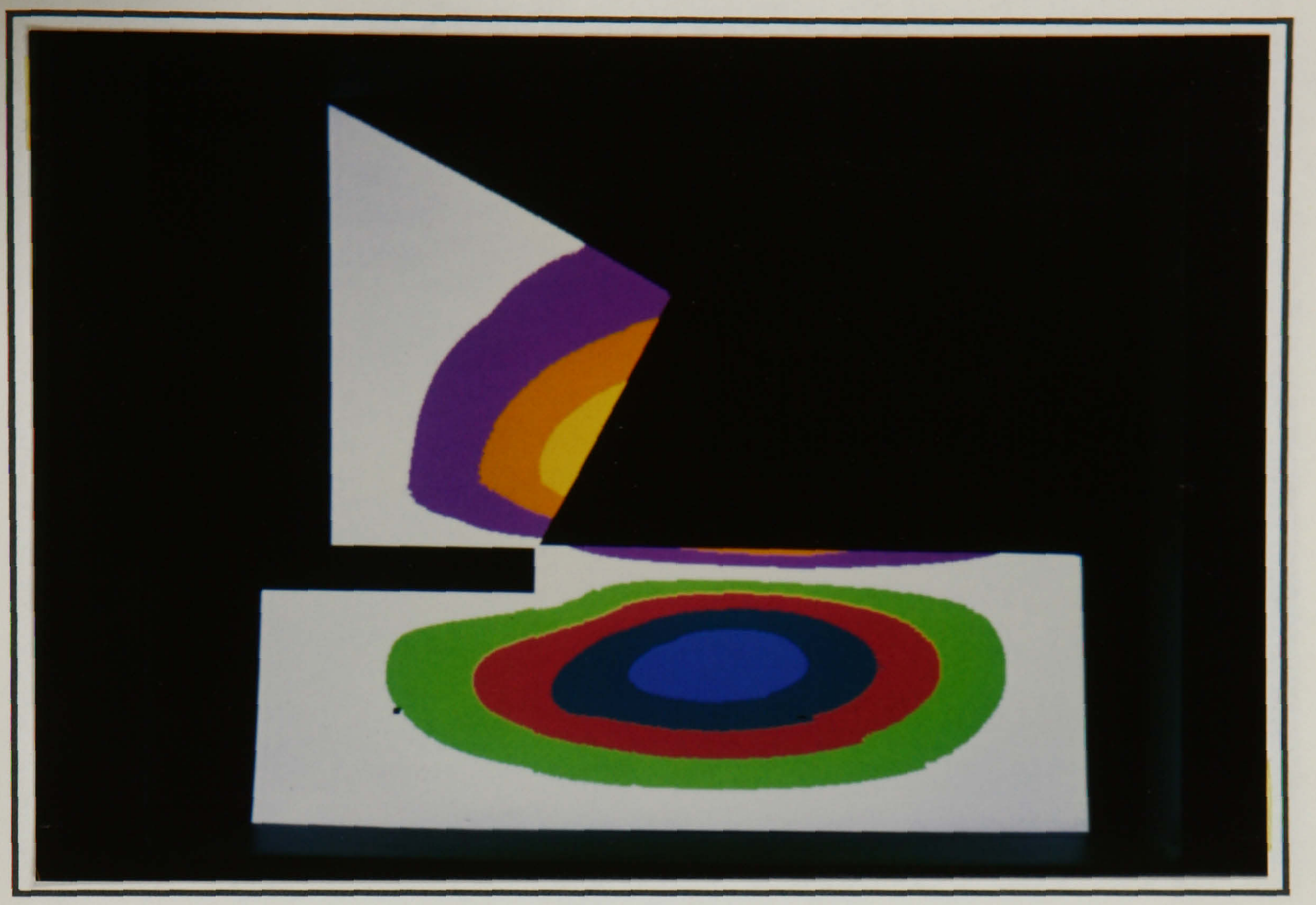
Photograph 7.29



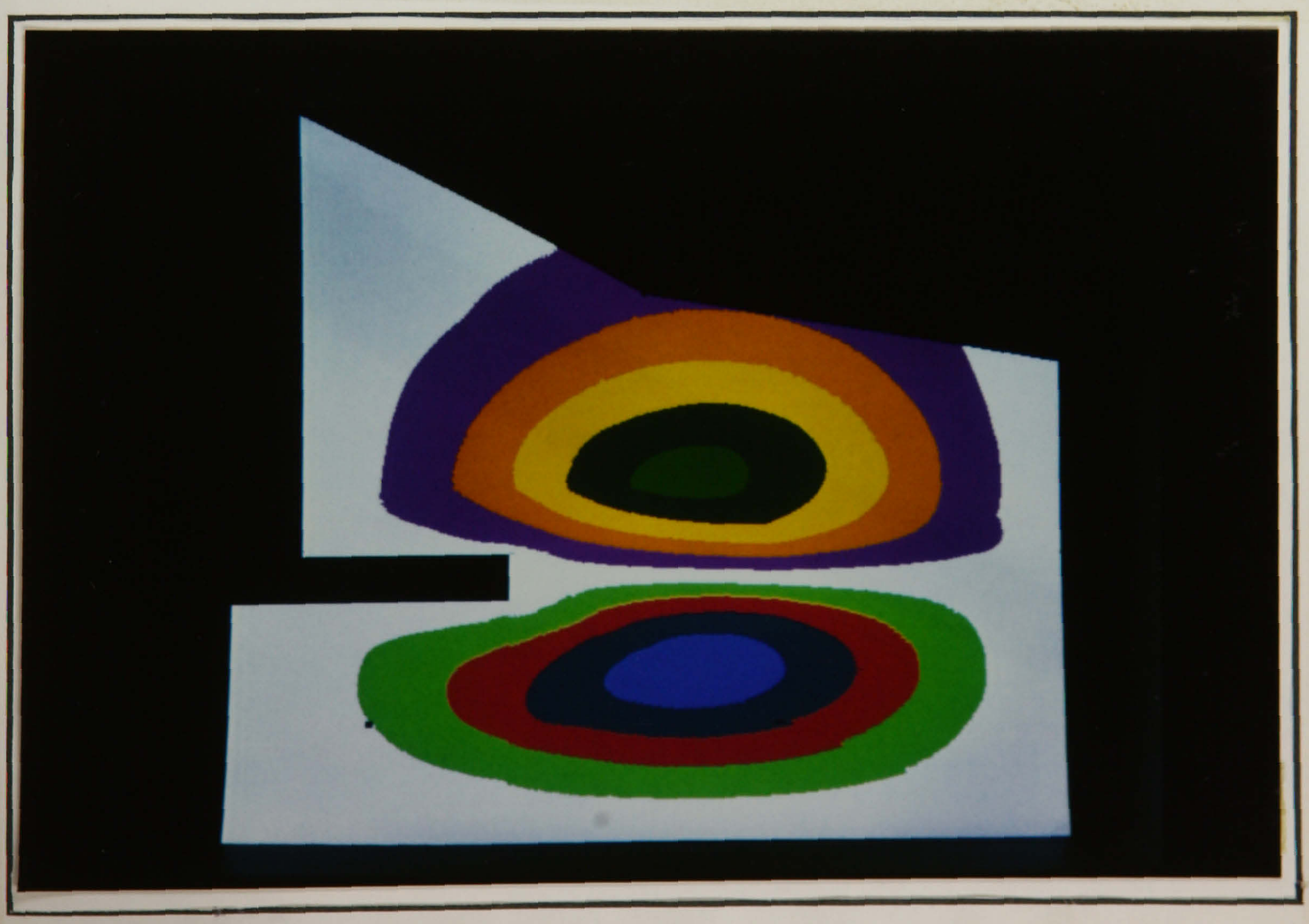
Photograph 7.30



Photograph 7.31



Photograph 7.32



Photograph 7.33

Photograph 7.34

7.5.2.2 **Cylinder rotor geometry.**

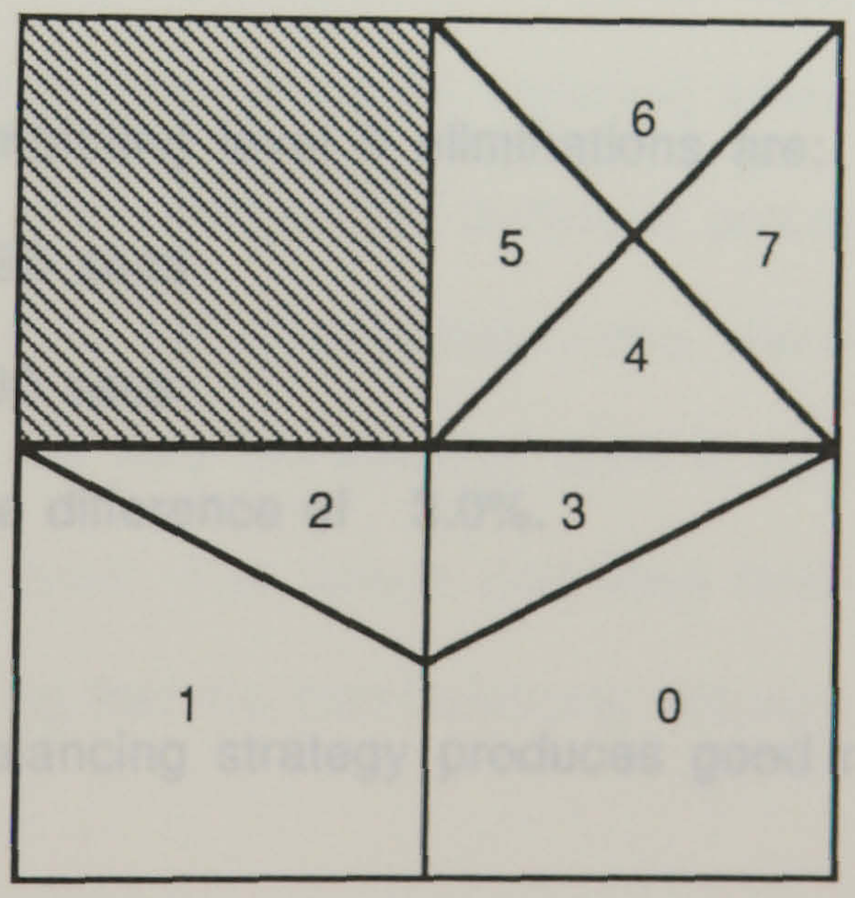


Fig. 7.35

Sectors 3 and 4 are of type C, sectors 0 and 1 are of type B and sectors 2,5,6,7 are of type D. Therefore sectors of type C are the most constrained. In sector C let the number of rows be 5 and the number of spokes be 9. The interdomain interface lies on Sc(1).

Now calculate the work load on C (Wc) from (7.9)

Wc = 163 * 439 = 71557

Using (7.13) calculate Sb(1) and Sb(2).

$$(Sb(1) + Sb(2)) = \frac{-1766 + 10346.8}{700} = 12.25$$

Sb(1) + Sb(2) = 12 (rounded)

Sb(1) = 6 and Sb(2) = 6

Wb = 128 * 539 = 68992

Using (7.15) calculate Sd(1).

$$Sd(1) = \frac{-2473 + 10347.99}{700} = 11.25$$

Sd(1) = 11 (rounded)

Wd = 128 * 539 = 68992

Experimental timings after completing level 0 eliminations are:

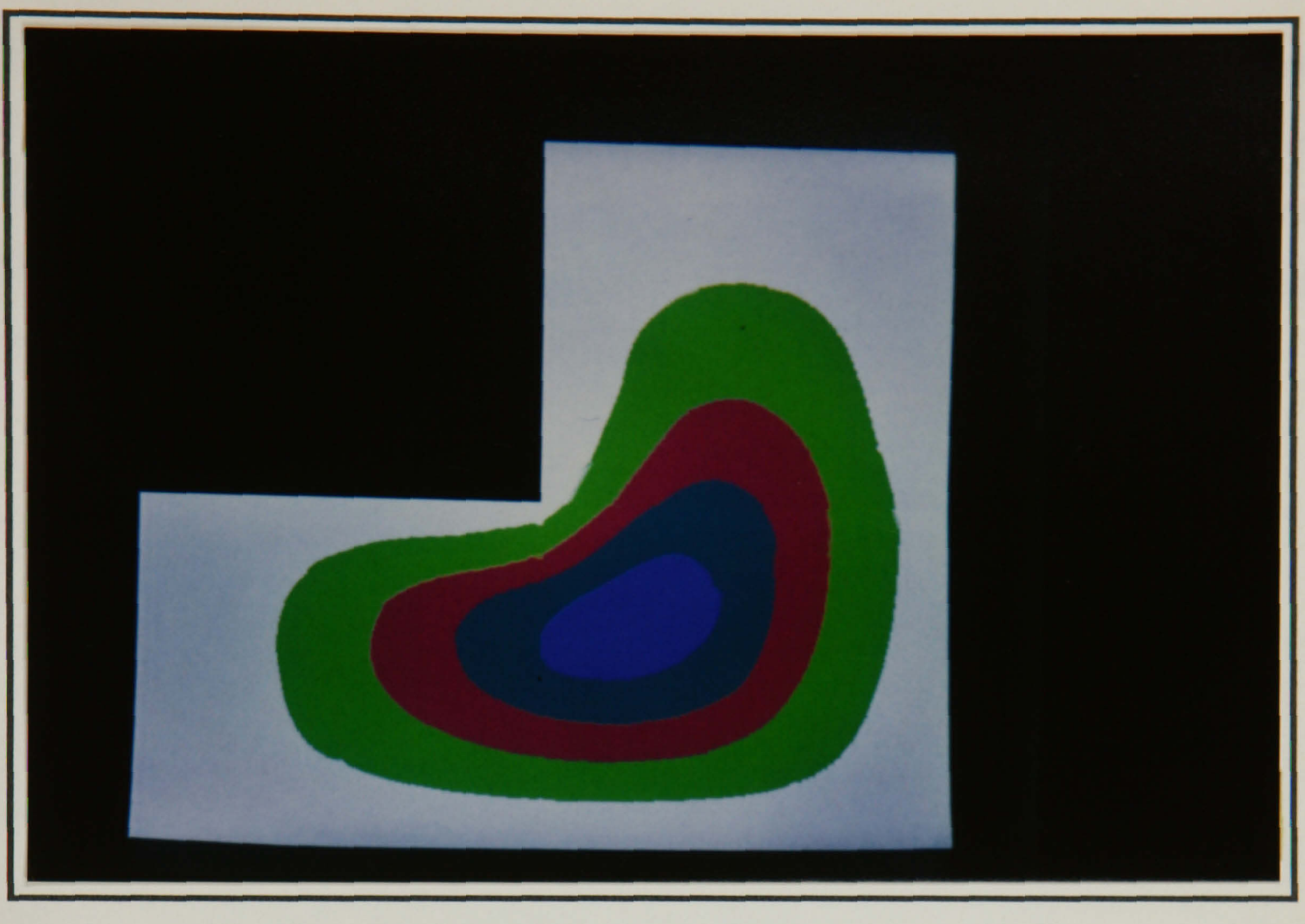
First transputer back: 234.096 secs

Last transputer back: 240.580 secs

with an approximate percentage difference of 3.0%.

This signifies that the load balancing strategy produces good results with only a 3.0% discrepancy recorded.

A single photograph of the streamlines is presented for the flow domain as a whole, these contours agree with the work presented in chapter 5 produced on the Vax and with that of Bodalia [6].



Photograph 7.36

Both examples show very good load balancing, for different size problems a more typical value for the percentage difference is closer to 8% which again is very good. The main reason for this difference arises from the error incurred when rounding the calculated number of spokes causing the work load on the particular processors to be slightly out of balance with the rest. The time for assembling extra elements adds a very small contribution to this difference and also the strategy used is only an approximation to the actual work load on each processor. The results that have been obtained from using the strategy are encouraging, showing that the load balancing strategy is an acceptable one.

7.6 Summary

An efficient parallel algorithm has been developed for both 2-D and 3-D axisymmetric flow problems, based on the multifrontal method. In our scheme the domain of the problem is sub-divided into a number of subdivisions and similar finite element meshes generated for each subdivision to guarantee load balancing over the processors. The data for each subdivision is then sent to the appropriate processor in the network.

The results obtained for all of the problems considered are in excellent agreement with our earlier work and the work of others. From the timings we have taken the algorithm makes efficient use of the processors and displays good speedup. The load balancing strategy that has been developed for non-convex domains achieves a satisfactory balancing across the processors.

CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

8.1 Summary And Conclusions

Investigational work has been carried out into the numerical simulation of fluid flow and mixing in a 2-D geometry and also in various 3-D axisymmetric geometries, for Newtonian and variable-viscosity fluids. The primitive variable finite element approach was found to be the most widely used finite element scheme for this class of fluid flow problem. Initial work was performed on a traditional Von Neumann architecture, sequential computer and later work involved writing a concurrent algorithm for a parallel architecture machine.

8.1.1 Plate Over A Slot

Two slightly different primitive variable (U,V,P) finite element schemes, Hughes and Taylor [42] and Crochet et al [19], were mentioned in the literature and both schemes were successfully implemented. We applied both formulations to a two dimensional test problem, driven flow over a square cavity, for both Newtonian and variable viscosity fluids and the results produced were almost identical when we compared both numbers and streamline plots. This indicated that neither scheme has advantages over the other and so there was no particular preference as to which scheme should be used.

For the Newtonian fluid the results were obtained for a range of values of Reynolds numbers from Re=1 to 1000, so that they could be compared with those presented in papers by Cliffe et al [15] and Mizukami [75]. The meshes used were identical to two of the coarser meshes used by Cliffe et al. The results presented in Cliffe et al's report are

for one of the more refined meshes, yet our results are in very good agreement and follow the same trends as Cliffe et al. Results were also obtained for the variable-viscosity fluid at a selection of Re values up to Re=250.

The finite element program uses a substantial amount of computation time, for this reason ways of improving the efficiency of our algorithm were considered. The most significant improvement was achieved by employing Newton's method instead of the straightforward linearisation technique previously used and the number of iterations to reach convergence was greatly reduced on employing Newton's method.

Some theoretical investigation of colour band and dispersive mixing was performed for the cavity geometry. The concentration of colour in the fluid is calculated as it changes with time until a homogeneous mixture is obtained. This is achieved by solving the convection/diffusion partial differential equation. We implemented a semidiscrete formulation incorporating the streamline upwind Petrov/Galerkin approach for the solution of this time dependent convection/diffusion equation. Good results were obtained for low Reynolds numbers, up to Re=10, but problems were encountered at higher Reynolds numbers. We believe the reason for this was caused by the singularities in the U velocity at the top corners of the cavity.

This work gave us a good grounding in the finite element method as applied to fluid flow problems, and a number of reliable programs were written which were rigorously tested and were shown to produce very good results. On this solid foundation we were able to proceed with confidence to investigate three dimensional axisymmetric flow problems.

The three dimensional axisymmetric flow problem meant that we now had four coupled partial differential equations to solve as opposed to three in the two dimensional case. The Hughes and Taylor U,V,P approach for the 2-D case was extended to a U,V,W,P approach for this problem. Three different mixing geometries were investigated for Newtonian and variable-viscosity type fluids.

8.1.2 **Cylinder Rotor**

Results were initially obtained for the Newtonian fluid at values of Reynolds numbers between Re=1 and 1000 inclusive, for the short cylinder geometry. A fairly refined mesh was required to obtain results up to Re=1000, and a streamline plot at Re=1000 showed the appearance of axisymmetric vortex breakdown. These results compared very well with those of Bodalia [6] up to Re=200. Results for the variable-viscosity fluid were then obtained for Re=1 to 500 and these also agreed very well with Bodalia as far as Re=100 . Both streamline and rV plots were compared and found to be in close agreement. Furthermore, it was noted that the finite difference program failed to reach the higher Reynolds numbers.

The short cylinder geometry was again used for the Newtonian fluid with the height of the vessel specified as five. This problem has the properties of a Couette type flow and results were found to show Taylor vortices appearing at approximately Re=60 as expected and results were obtained up to Re=100. These results were compared with Neitzel [82] and with results produced by D.G. Knight running a version of Bodalia's finite difference program, and were found to be in good agreement. The plots showed the phenomenon of Taylor vortices very well, i.e. the appearance of contra-rotating vortex pairs of equal magnitude. The same geometry was then used for the variable-viscosity fluid obtaining results at Re=1 and also at values up to Re=80. Here Taylor vortices appeared earlier as expected, because of the shear thinning nature of the fluid.

For the short cylinder geometry some investigational work was undertaken into colour band mixing for time dependent steady flows. A semidiscrete finite element/finite difference formulation was employed to solve the problem and it was also necessary to incorporate upwinding into the formulation for instances when the flow was advection dominated. For the cylinder geometry results were obtained for Reynolds numbers higher than Re=100 without upwinding, and with upwinding we were able to produce excellent results at Re=800 even though by this stage the non-upwinded results had become totally corrupted and extremely unstable.

8.1.3 Disc Rotor

Next we investigated the flow induced by a disc stirrer rotating in a cylindrical vessel. Results for Newtonian fluids were calculated for a range of Reynolds numbers starting at Re=1 up to Re=500 even though results at higher Re numbers were attainable. For the variable-viscosity fluid results were found for Reynolds numbers up to 250. These results were then compared with Bodalia [6] and Spragg [103] and again were found to be in very good agreement.

Some investigational work was also carried out into colour band mixing for the disc rotor geometry for a non-Newtonian fluid and our results proved to be very similar to those of Spragg. From the excellent results produced by the using the upwind method our confidence in the merit of the method was reaffirmed.

8.1.4 Axisymmetric Rushton Turbine

The final geometry to be considered was the axisymmetric Rushton turbine or flat blade paddle problem which was the most complicated and also the more realistic geometry of the three geometries. The results were compared with the work of Voncken et al [111], which showed broad similarity with our work.

It is worth mentioning again the remarkable advantage of the finite element method in that the program that had been written for the cylindrical stirrer did not have to altered at all for the other two geometries. Only a new data file describing the new geometries had to be generated. This is very different to finite difference work where it would be necessary to make a large number of modifications to the finite difference program when changing the geometry of the problem.

Another important point is that the complex Rushton geometry did not present any problem for the finite element method, and even more complex geometries could be easily dealt with.

8.1.5 Work carried out on a multiprocessor local memory system.

It was noticed that the finite element program was computationally very costly and fairly slow. Therefore, consideration was given to finding ways of speeding up the program. For the 2-D cavity problem we implemented Newton's iterative method which greatly reduced the number of iterations required to reach convergence. Due to time limitations we were unable to implement it for the 3-D case even though a similar improvement was expected.

This problem of computational time and memory storage is of paramount importance when considering the very complex visco-elastic fluid flow problem. In this type of problem the number of equations to solve would be increased from four to ten, leading to the number of degrees of freedom per element being greatly increased, which inevitably would cause a significant increase the size of the frontwidth.

To attempt to solve a problem of this magnitude the benefit of a small reduction in computation time gained by employing Newton's method was not enough, the only realistic solution being to transfer to a more powerful computer, either a super-computer or a parallel architecture machine. We decided to work on a multi-processor system consisting of a number of transputers. Since this was a research topic in which there was much current interest, and remains so, a significant contribution would therefore be made by simulating fluid flow problems on this type of system.

The remainder of the research time was spent in designing, developing and testing an efficient parallel algorithm based on the multifrontal method. The program was applied to a few of the 2-D and 3-D axisymmetric Newtonian flow problems described in chapters 4 and 5. In the approach we employed the domain of the problem is partitioned into a number of parts and similar finite element meshes generated in each partition in order to achieve load balancing. The information relating to each partition is then placed on separate processors in the network, where the number of processors in the network equals the number of partitions. The load balancing is naturally satisfied for a problem with a convex

domain, but for the case of a non-convex domain a load balancing strategy was developed which balances the load over the processors very well.

In testing the program for our particular problems we obtained results which agreed well with our earlier work. Our results also demonstrate that the method utilises the processors efficiently and good speedup is possible from the system. This indicates that with more processors in the network the computation time could be significantly improved and even larger problems considered. On successfully arriving at this milestone in our research the outlook for future work on viscoelastic flows seems very favourable.

From this research work three papers [71,72,73] have been published so far and are contained in the appendix.

8.2 Recommendations For Future Work

On the basis of work carried out we thus have a number of ideas for further research work to give a better understanding of mixing flows :-

(i) The calculation of the power consumption of the mixing system, which is important in industrial applications where the knowledge of the most efficient mixing system together with the least power input is required. It is also important to calculate the couple (that is the power) on the stirrer and the pumping number (that is power consumed). This involves a small amount of work given the flow field e.g. Bodalia [6].

(ii) The study of various stirrers rotating in visco-elastic liquids which are very complex fluids and are difficult to implement computationally to obtain meaningful results for reasonable values of elasticity. This could be achieved by modifying our parallel algorithm to incorporate the new mixed finite element method developed by Marchal and Crochet [68].

(iii) A study of transient flows, that is from start-up which would include time-dependent terms in the equations.

(iv) Implement some form of free surface strategy to achieve a more realistic model, see [19] and [78].

(v) Theoretical investigation of the temperature distribution in the fluid as it changes with respect to time. This is achieved by solving the temperature equation, similar to the concentration equation presented in chapters 4 and 5.

(vi) Theoretical investigation of turbulence in mixing flows adopting the new mathematics of Chaos.

REFERENCES

[1] AMESTOY P. and TILCH R.

Impact Of Comput. In Sci. and Engrng., 1989, 1, 93-107.

[2] BARNES H.A.

Chapter 2,

Rheometry: Industrial Applications,

ed. K. Walters, Research Studies Press, 1980.

[3] BARRAGY E. and CAREY G.F.

Int. J. For Num. Meth. In Engrng., 1988, 26, 2367-2382.

[4] BARTELA M. and GORI F.

J. of Fluids Eng. 1982, 104, 31

[5] BIRD R.B., ARMSTRONG R.C. and HASSAGER O.

Dynamics Of Polymeric Liquids, Volume 1 Fluid Mechanics,

John Wiley and Sons, 1977

[6] BODALIA V.

PhD Thesis 1986

Depts. of Chem. Eng. and Maths and Computer Science

Poly. of Wales

[7] BORNE J.R.

Brit. J. Appl. Phys. 1965, 16, 1411

[8] BOWLER K.C., KENWAY R.D., PAWLEY G.S. and ROWETH D.

An Introduction to Occam2 Programming

Chartwell-Bratt Publishing Ltd, Sweden, 1987

[9] BOZEMAN J.D. and DALTON C.

J. of Comp. Phys. 1973, 12, 348-363

[1 0] BROOKES G.R. and STEWART A.J.

Introduction to OCCAM 2 on the Transputer

Macmillan Education Ltd., 1989.

[11] BROOKS A.N. and HUGHES T.J.R.

Comput. Meth. Appl. Mech. Engrg. ,1982, 32, 199-259.

[12] BURNS A.

Programming in OCCAM 2

Addison-Wesley Publishers Ltd., 1988.

[13] CARLING A.

Parallel Processing - The Transputer and Occam

Sigma Press, 1988.

[14] CHIEN W.-L., RISING H. and OTTINO J.M.

J. Fluid Mech. 1986, 170, 355

[15] CLIFFE K.A. et al

Theoretical Physics Division

UK AERE Harwell Rep. R.9202, 1978

[16] CLIFFE K.A. and MULLIN T.

J. Fluid Mech. 1985, 153, 243-258

[17] COLLIAS D.J. and PRUD'HOMME R.K.

Chem. Eng. Sci. 1985, 40, 1495

[18] COULSON J.M. and RICHARDSON J.F.

Chemical Engineering, Vol. 2, Third Edition

Permagon Press, Oxford, 1978

[19] CROCHET M.J., DAVIES A.R. AND WALTERS K.

Numerical Simulation of Non-Newtonian Flow

Rheology Series 1, Elsevier 1984

[20] CROSS M., RICHARDS C.W., IEROTHEOU C. and LEGGETT P.

Chapter 5, Flow Modelling In Industrial Processes,

Ellis Horwood Limited, 1989.

[21] CUTHILL E. and MCKEE J.

Proc. 24th National Conf. A.C.M., 1969, 157-172

[22] DAVIES A.J.

The Finite Element Method: A First Approach

Claredon Press Oxford 1980

[23] DHATT G., HUBERT G., NGUYEN D. and THIOLLET C.

Proceedings of the Fourth International Conference on

Numerical Methods in Laminar and Turbulent Flows

Vol. 1, C. Taylor et al (editors), Pineridge Press,

Swansea, 1985, 107-119

[24] DIJKSTRA D. and VAN HEIJST G.J.F.

J. Fluid Mech. 1983, 128, 123

[25] DUFF I.S.

Harwell Report. CSS 89. 1981.

[26] DUFF I.S.

Harwell Report. AERE R/10079. 1981.

[27] DUFF I.S. and REID J.K.

SIAM J. Sci. Stat. Comput., Sept. 1984, 5, 633-641.

[28] DUFF I.S.

Parallel Computing, 1986, 3, 193-204.

[29] DUFF I.S.

Harwell Report. CSS 210. 1988.

[30] ESCUDIER M.P.

Exp. Fluids 1984, 2, 189

[31] FARHAT C. and WILSON E.

Int. J. For Num. Meth. In Eng., 1987, 24, 1771-1792.

[32] FOSDICK K.L. and KAO B.G.

Rheol. Acta. 1980, 19, 675

[33] GIESEKUS H.

Rheol. Acta. 1967, 6, 339

[34] GREENOUGH C. and ROBINSON K.

NAg Finite Element Library

Level 0 and Level 1 Documentation

Rutherford Appleton Laboratory 1982

[35] GRIFFITHS D.I., JONES D.T. and WALTERS K.

J. Fluid Mech. 1969, 36, 161

[36] GRIFFITHS D.I. and WALTERS K.

J. Fluid Mech. 1970, 42, 379

[37] HARNBY N., EDWARDS M.F. and NIENOW A.W. (editors)

Mixing in the Process Industries.

Butterworth and Co. (Publishers) Ltd, 1985

[38] HILL G.

Transputer Networks using the IMSB003

Technical Note 13. INMOS 1987.

[39] HOARE C.A.R.

Communicating Sequential Processes

Prentice-Hall, 1985.

[40] HOLLAND F.A.

Fluid Flow,

Reinhold, New York, 1966

[41] HOOD P.

Int. J. For Num. Meth. In Eng., 1976, 10, 379-399

[42] HUGHES T.G. and TAYLOR C.

Finite Element Programming of the Navier-Stokes,

Equations, Pineridge Press Limited, Swansea, 1981.

[43] HUGHES T.J.R. , FRANCA L.P. and MALLET M.

Comput. Meth. Appl. Mech. Engrg. ,1986, 54, 223-234.

[44] HUGHES T.J.R., MALLET M. and MIZUKAMI A.

Comput. Meth. Appl. Mech. Engrg. ,1986, 54, 341-355.

[45] HUGHES T.J.R. and MALLET M.

Comput. Meth. Appl. Mech. Engrg. ,1986, 58, 305-328.

[46] HUGHES T.J.R. and MALLET M.

Comput. Meth. Appl. Mech. Engrg. ,1986, 58, 329-336.

[47] HUTTON A.G.

Proceedings of the fourth International Conference

on Laminar and Turbulent Flows.

Pineridge Press, Swansea, 1985, 289-305.

[48] HYMAN D.

Advances in Chemical Engineering : Volume 3

-Mixing and Agitation

Academic Press 1962, 119-202.

[49] HYUN J.M.

Trans. of the ASME, 1985, 107, 92.

[50] INMOS Ltd.

Transputer Developement System

Prentice-Hall, 1988.

[51] INMOS Ltd.

Transputer Instruction Set: A Compiler Writer's Guide

Prentice-Hall, 1988.

[52] INMOS Ltd.

Transputer Reference Manual, Prentice-Hall, 1988.

[53] IRONS B.M.

Int. J. Num. Meth. in Eng. 1970, 2, 5-32.

[54] ISSA R.I. and GOSMAN A.D.

Proceeding of the Second International Conference

on Numerical Methods in Laminar and Turbulent Flows

Pineridge Press, 1981, 827.

[55] JEFFREY G.B.

Proc. London Maths Soc. 1915, 14, 327.

[56] JOHNSON C.

(eds. R. Glowinski and J.L. Lions),

Computing Meth. in Engrng. and Appl. Sci. Vol V,

North-Holland, 1982.

[57] JOHNSON C.

Finite Elements in Fluids, Vol. VI, Wiley, London, 1985.

[58] JOHNSON C., NAVERT U. and PITKARANTA J.

Computer Meth. in Appl. Mech. and Engrng, 1984, 45, 285-312.

[59] JOHNSON C. and SARANEN J.

Mathematics of Computation, 1986, 47, 175, 1-18.

[60] KALE D.D., MASHELKAR R.A. and ULBRECHT J.

Rheol. Acta. 1975, 14, 631.

[61] KEENTOK M., MILTHORPE J.F. and O'DONOVAN E.

J. of Non-Newtonian Fluid Mech. 1985, 17, 23.

[62] KERRIDGE J.

OCCAM Programming: A Practical Approach

Blackwell Scientific Publications, 1987.

[63] KRAMER J.M. and JOHNSON M.W.

Trans. Rheol. 1972, 16, 197.

[64] KURIYAMA M., INOMATA H., ARAI K. and SAITO S.

AIChE 1982, 28, 385.

[65] LAN A. and ANGELINO H.

First European Conference on Mixing and Centrifugal

Seperation, Coles N.G. (editor), Bedford, B.H.R.A., 1974.

[66] LOHNER R.

Int. J. For Num. Meth. In Engrng., 1987, 24, 1741-1756.

[67] LUGT H.J. and ABBOUD M.

J. Fluid Mech. 1987, 179, 179.

[68] MARCHAL M. and CROCHET M.J.

J. Of. Non-Newt. Fluid Mech., 1987, 26, 77-114

[69] MARKOWITZ H.M.

Management Science, 1957, 3, 255-269.

[70] MELHEM R.G.

University of Pittsburgh

Technical Report ICMA-85-84, 1985

[71] MILES R.G. and HAVARD S.P.

Chapter 3, Flow Modelling In Industrial Processes,

Ellis Horwood Limited, 1989.

[72] MILES R.G. and HAVARD S.P.

Int. J. For Num. Meth. In Fluids, 1989, 9, 731-740.

[73] MILES R.G. and HAVARD S.P.

Proceedings of the Sixth International Conference

on Numerical Methods in Laminar and Turbulent Flows ,

Pineridge Press, Swansea, 1989

[74] MITCHKA P. and ULBRECHT J.

Appl. Sci. Res. 1966, 15, 345.

[75] MIZUKAMI A.

Proceedings of the Fourth International Conference on

Numerical Methods in Laminar and Turbulent Flows

Vol. 1, Pineridge Press, Swansea, 1985, 140-151.

[76] MODI J.J.

Parallel Algorithms and Matrix Computation

Clarendon Press, Oxford, 1988.

[77] MULLIN T. and LORENZEN A.

J. Fluid Mech. 1985, 157, 289-303.

[78] NAGATA S.

Mixing: Principles and Applications

John Wiley and Sons, London, 1975

[79] NALLASAMY M. and KRISHNA PRASAD K.

J. Fluid Mech. 1977, Vol. 79, part 2, 391-414

[80] NASSER A.G.F.A. and LESCHZINER M.A.

Proceedings of the Fourth International Conference

on Numerical Methods in Laminar and Turbulent Flows- Vol. 1

Pineridge Press, Swansea, 1985, 480-491.

[81] NAVERT U.

Ph.D. Thesis, Department of Computer Science,

Chalmers University of Technology, Goteborg, Sweden,1982.

[82] NEITZEL G.P.

J. Fluid Mech. 1984, 141, 51-66.

[83] NIRSCHL J.P. and STEWART W.E.

J. of Non-Newtonian Fluid Mech. 1984, 16, 233.

[84] NOUR-OMID B., RAEFSKY A. and LYZENGA G.

A.M.D., 1987, 86, 209-227.

[85] OLDSHUE J.Y.

Fluid Mixing Technology.

McGraw-Hill Publications Company, New York, 1983

[86] ORTEGA J.M. and VOIGT R.G.

Solution Of Partial Differential Equations on Vector and

Parallel Computers. SIAM, Philadelphia, 1985.

[87] OTTINO J.M.

The kinematics of mixing: stretching, chaos, and transport

Cambridge University Press, 1989

[88] PADDON D.J.

PhD Thesis U.C. Wales 1980

[89] PADDON D.J. and WALTERS K.

Rheol. Acta. 1979, 18, 565.

[90] PAO H-P.

The Physics of Fluids, 1972, 15, 4.

[91] PEETERS M.F., HABASHI W.G. and DUECK E.G.
Proceedings of the Fourth International Conference on
Numerical Methods in Laminar and Turbulent Flows
Vol. 1, Pineridge Press, Swansea, 1985, 433-444.

[92] PEYRET R. and TAYLOR T.D.
Comp. Methods of Fluid Flow, Springer, 1983.

[93] POLYFLOW;
Prof. M.J. CROCHET, Unite de Mecanique Appliquee,
University Louvain-la-Neuve, Belgium.

[94] POUNTAIN D. and MAY D.
A Tutorial Introduction To Occam Programming
Blackwell Scientific, 1987.

[95] QUILLEN C.S.
Chem. Eng., 1954, 61, 6.

[96] QUINN M.J.
Designing Efficient Algorithms For Parallel Computers
McGraw-Hill, 1987.

[97] REDDY J.N.
An Introduction To The Finite Element Method
McGraw-Hill Book Company, Singapore, 1984

[98] REID J.K.
Harwell Report. CSS 155. 1984.

[99] ROACHE P.J.
Computational Fluid Dynamics
Albuquerque, Hermosa Publishers, 1976.

[100] RUSHTON J.H.
Ind. Eng. Chem., 1955, 47, 582.

[101] SAAD Y. and SCHULTZ M.
Research report. YALE DCS RR-461,
Dept. of Computer Science, Yale University, 1986.

[102] SMITH G.D.

Numerical Solution of Partial Differential Equations:

Finite Difference Methods, Second edition

Claredon Press, Oxford, 1979.

[103] SPRAGG A.J.P., MASKELL S.J. and PATRICK M.A.

Proceedings of the Fourth International Conference

on Numerical Methods in Laminar and Turbulent Flows - Vol. 1

Pineridge Press, Swansea, 1985, 39-50.

[104] STUART J.T.

Taylor-Vortex Flow: A Dynamical System

SIAM Review, Sept. 1986, 28, 315-342.

[105] THOMAS R.S. and WALTERS K.

Q. Jl. Mech. Appl. Math. 1964, 17, 39.

[106] ULBRECHT J.

Chem. Eng. 1974, 347.

[107] ULBRECHT J. and GASPARETTO

J. Rheol. 1980, 24, 551.

[108] ULBRECHT J. and PATTERSON G.K.

Mixing of Liquids by Mechanical Agitation

Gordon and Breach Science Publishers, 1985.

[109] VAN WAZER J.R. et al

Viscosity and Flow Measurement

Interscience, London, 1963.

[110] VOGEL H.U.

Max-Planck-Institut Fur Stromungsforschung,

Gottingen, Bericht 6, 1968

[111] VONCKEN R.M., ROTTE J.W. and TEN HOUTEN A. TH.

A.I.Ch.E-I.Chem.E. Symposium Series No. 10, 1965.

[112] VON KARMAN T.H.

Z.A.M.M. 1921, 1, 233.

[113] WAIT R. AND MITCHELL A.R.

Finite Element Analysis and Applications

Wiley-Interscience Publication, 1985.

[114] WALTERS K.

Rheometry

Chapman and Hall, London, 1975.

[115] WALTERS K.

J. Fluid Mech. 1970, 40, 191.

[116] WALTERS K. and WATERS N.D.

Brit. J. Appl. Phy. 1963, 14, 667.

[117] WALTERS K. and WATERS N.D.

Polymer Systems - Deformation and Flow

Macmillan, London, 1968.

[118] WANG J.H. and YANG W.J.

Proceedings of the second International conference

on Laminar and Turbulent Flow, 1983.

[119] WATERS N.D. and GOODEN D.K.

Q. Jl. Mech. Appl. Math. 1980, 33, 190.

[120] WATERS N.D. and KING M.J.

Q. Jl. Mech. Appl. Math. 1971, 24, 331.

[121] WEIN O.

J. of Non-Newt. Fluid Mech. 1976, 1, 357.

[122] WEIN O.

Rheol. Acta. 1977, 16, 248.

[123] WILLIAMS E.W.

J. of Non-Newt. Fluid Mech. 1976, 1, 51.

[124] WILLIAMS R.D. and GLOWINSKI R.

Proceedings of the Sixth International Conference

on Numerical Methods in Laminar and Turbulent Flow.

Pineridge Press, Swansea, 1989.

[125] WILLIAMS R.W.

Ph.D. Thesis U.C. Wales 1979

[126] WILLIAMS R.W.

Rheol. Acta. 1979, 18, 345.

[127] WILLIAMS R.W.

Rheol. Acta. 1980, 19, 448.

[128] WINTERS K.H. and CLIFFE K.A.

UK AERE Harwell Rep. R.9444, 1979.

[129] ZIENKIEWICZ O.C.

The Finite Element Method.

Third Edition, McGraw-Hill, 1977

APPENDICES

APPENDIX

I

A paper presented at the Polymodel conference 1988, discussing preliminary work undertaken on a transputer system for solving fluid mechanical problems using the finite element method and published in the proceedings of the conference.

CHAPTER 3

TRANSPUTER ARRAYS FOR SOLVING FINITE ELEMENT SYSTEMS

R.G. Miles and S.P. Havard
Department of Mathematics & Computer Science
The Polytechnic of Wales

Introduction

When solving finite element problems on a workstation the solution may take a considerable time to complete. To significantly decrease this time either very much faster processors must be used or a new approach adopted. In this paper we consider the feasibility of a new approach, that of using a network of fast processors to solve the problem. This paper is subdivided into four sections:

(i) a discussion of the transputer, transputer networks and the types of constraint that this imposes on algorithm design;
(ii) an outline of frontal and multifrontal solvers;
(iii) a section showing how these methods may be used on transputer networks;
(iv) tests will be given for two problems

  (a) Laplace's equation in a square region with Dirichlet boundary conditions
  (b) a driven flow over a slot;

(i) Transputer Networks

The transputer is a computer in silicon. Each chip contains a processor, memory and communication links. The transputers that we have used are of two types. The T414 has a 32-bit microprocessor, 4kbytes of on-chip RAM and four standard communication links. All floating point operations are done via software however, for the new transputer, T800, a 64-bit floating-point unit is also included on the chip. The four interface communication links allow for communication between transputers on a point-to-point basis. Since the links and the processor are largely independent the processor can continue working whilst the transputer is passing messages via the links. By passing messages information may be transferred from one transputer to another and consequently a network of transputers may be constructed.

Various building blocks for constructing transputer networks are now available, the most common for small systems being the 4-transputer board. The board that we have used has 4-transputers each with 1 Mbyte of memory attached and two of the links on each are wired to adjacent transputers, however, eight links remain unconnected. This network needs to communicate with the outside world and so another transputer is used as the root linking with a host computer. The most common host is an IBM-PC and these boards fit into the expansion slots, however, a number of other hosts are available, e.g. SUN. So the simplest network is as shown in Fig. 1.

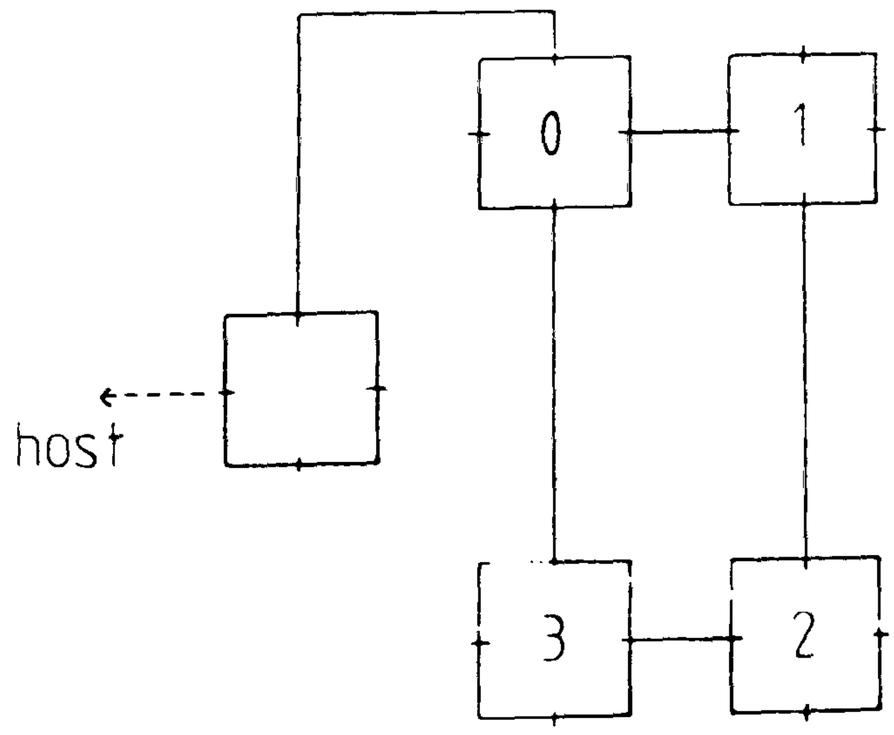


Figure 1
A simple 4-transputer network with root to host

More transputers may need to be configured into the network. This is achieved by daisy chaining the 4-transputer boards together and then using the unconnected links to give the desired configuration. The system may be configured as a ring, array or modified hypercube.

The results in the last section will be presented for networking four and eight transputers in a ring configuration. This is a two-directional ring and it is very straightforward to route messages in either the clockwise or the anti-clockwise directions so that they get to the required processors in the shortest time. This ring is shown in Fig. 2.

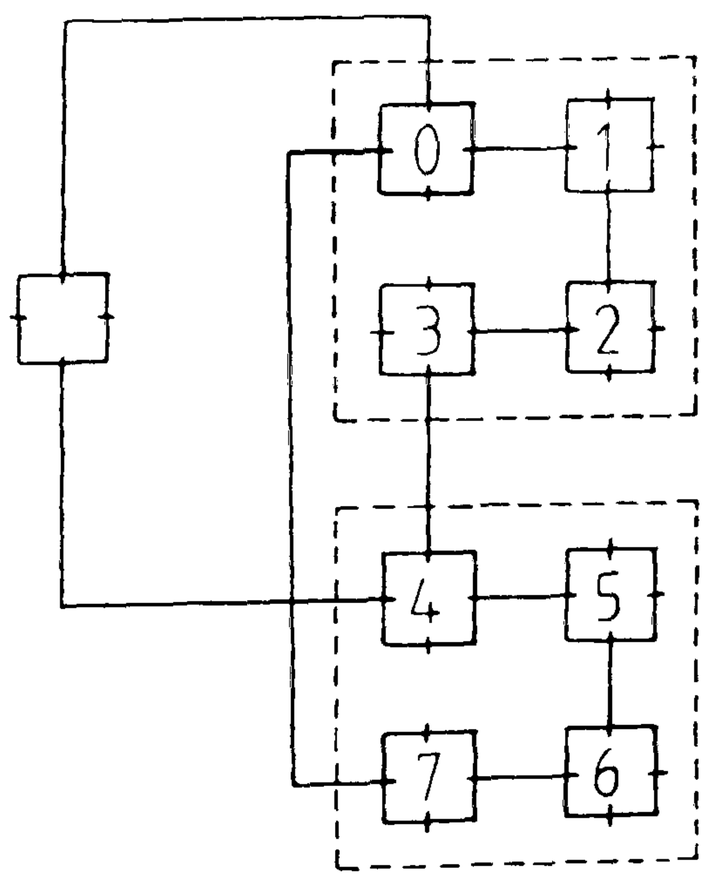


Figure 2
A configuration of 8 transputers in a ring with a root

For networks of transputers exceeding eight this type of configuration is only useful if there is a very small amount of communication or if communication is only with nearest neighbours. For example with sixteen transputers, the diameter of the network is eight, however, a diameter of four is achieved by using a two-dimensional array. There is an INMOS technical note on configurations, Hill(1).

The important characteristics of these networks are:

(a) each transputer has its own memory of 1Mbyte or more;
(b) each transputer is controlled by its own set of instructions although, in practice, a majority of transputers may be executing the same process;
(c) transputers communicate by passing messages and it is straightforward to pass messages from one transputer to another in the network;
(d) there may well be a large number of transputers in the network;
(e) there is no shared memory and no shared clock, consequently each transputer is a totally independent unit which can only communicate with other transputers by message passing.

(ii) Frontal and multifrontal schemes

The frontal scheme as proposed by Irons (2) is a modified Gauss Elimination procedure which is very frequently used in finite element problems. These problems have three important parts, namely the assembly of the equations, the elimination and the backsubstitution. Irons suggested that elimination should take place whenever possible rather than assembling the whole system and then performing eliminations. The method can be visualised using Fig. 3 assuming one degree of freedom per node.

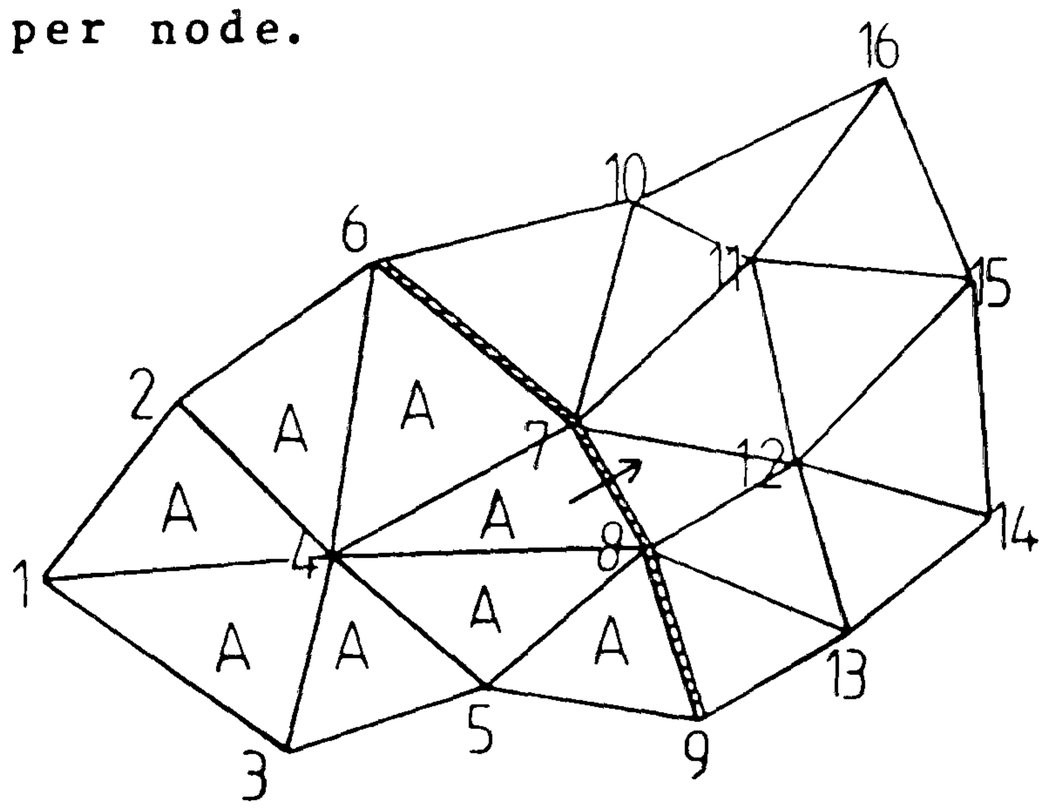


Figure 3
Finite element mesh with one front.

As the front progresses across the finite element mesh elements behind it will have been assembled and contributions made to the system matrix. The nodes not on the front will have had all of their contributions completed and so the rows associated with these nodes will have been eliminated. The elements ahead of the front have yet to be assembled and consequently their nodes have not yet entered the system. The basic frontal scheme may be summarised as having the following steps:

(a) assemble the matrix for an element;
(b) incorporate this into the system matrix;
(c) see if any rows correspond to fully contributed nodes; if so store the row, the variables associated with it and the right-hand side, and then eliminate the row;
(d) continue (a), (b) and (c) until all elements have been assembled. Then continue step (c) to complete the elimination;
(e) perform a back-substitution.

In practice a number of improvements have been made to this scheme, the most important of which, for non-positive definite matrices, is to try to retain accuracy by allowing the system matrix to buildup until it reaches a critical level where it will have a number of possible rows to eliminate. Some of these, enough to allow another element to be assembled, are eliminated using the largest pivots.

This scheme was modified by Hood(3) to treat unsymmetric matrices with symmetric sparsity pattern and by Duff(4),(5) who treated the unsymmetric patterns. Pivots are not necessarily diagonal elements and not all fully contributed variables are eliminated. Reid(6) suggested substructuring the finite element problem and used a tree to represent the order of assembly. There may be many frontal matrices and the method is called multifrontal. Duff & Reid(7) applied the multifrontal scheme to solve unsymmetric sets of linear equations. They use the flexibility that this introduces to make pivot choices which are economical in arithmetic operations. The core of the method is to build up an elimination tree to control the procedure. Duff(8) suggested that the elimination tree can act as a schedule for parallel computation and in (9) implemented a scheme on a shared memory multiprocessor.

(iii) Implementing Multifrontal schemes on Transputer Networks

There are three principal considerations that must be taken into account when implementing schemes on transputer networks. These are:

(a) granularity;
(b) load-balancing;
(c) an overall organisational data structure.

Communication can cause bottlenecks in networks so we want to achieve an algorithm with large granularity to minimise the amount of communication. The speed at which the scheme works is dependent on how efficiently each processor is used. If one transputer is given considerably more work to do than the others then this one will effectively control the network and the other processors will be idle, waiting for information. We require an algorithm which balances the work over the available transputers. The same main process is to be allocated to each transputer so a simple data structure is required to organise the algorithm especially the passing of messages from one transputer to another. The multifrontal scheme as outlined in (ii) seems very suitable with respect to the above considerations.

A pre-processing algorithm is used which runs on the root transputer to divide the finite element mesh over the number of available transputers and so allow a frontal, or multifrontal, scheme to work on each sector. In this study we have placed a modified frontal scheme on each transputer. In this way large granularity is achieved. With respect to load balancing we have chosen to reconsider the finite element mesh. We have only worked with meshes associated with convex regions and have used a hub and spokes. An example of this type of mesh for a square domain and an eight transputer system is shown in Fig. 4. The domain is divided into sectors dependent on the number of available transputers and then the finite element mesh is formed accordingly. Each transputer is sent the information relating to its sector along with information about the global structure of the mesh. The latter information is necessary when transputers pass messages to each other.

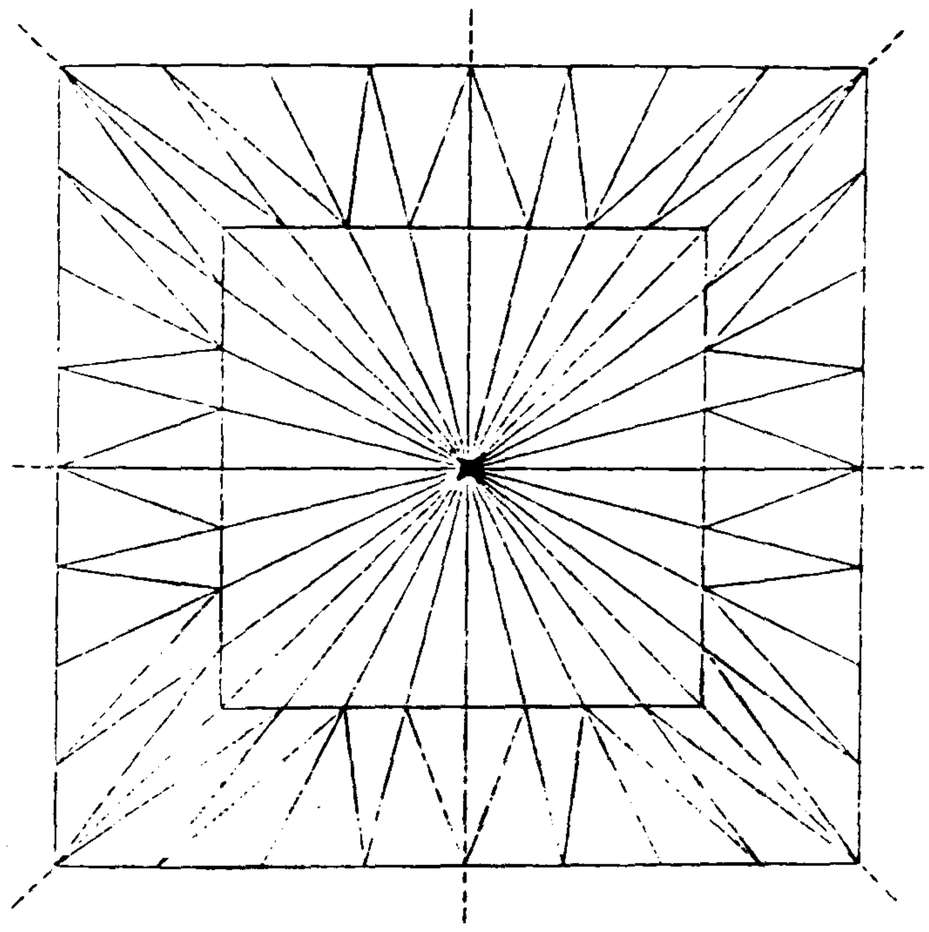


Figure 4.
A mesh for a square domain with eight transputers

Each transputer works on its own sector and when completed the matrices will be associated only with interface variables (i.e. those on the boundaries between sectors) and all other variables will have had their associated rows stored and have been eliminated. Information must then be passed between transputers and by combining matrices more variables have full contributions and so become available for elimination. The best organisational structure for this is a tree, see, Fig. 5.

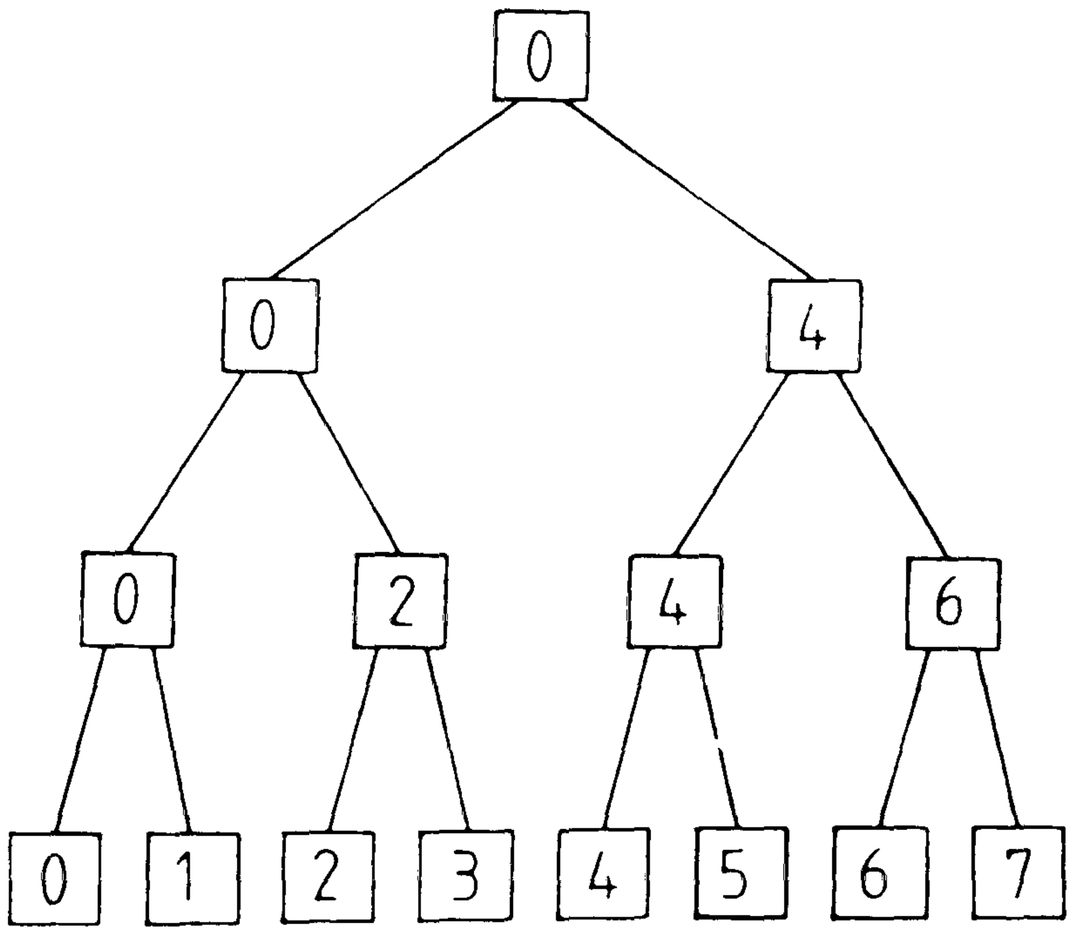


Figure 5
A tree structure for eight transputers

As the tree is traversed each transputer knows whether it is to receive or send data and is in a suitable state to do so. For the elimination phase the tree is traversed from the leaves towards the root and then in reverse for the backsubstitution.

(iv)Results

The first test problem for this algorithm was Laplace's equation in a square region with Dirichlet boundary conditions. We have used a triangular mesh with one degree of freedom per node, consequently the element matrix is extremely easy to form.

First of all the processor efficiency for the algorithm was examined. A test was undertaken using 15 rows, 11 spokes and 4 processors. This problem has 600 variables . Timings were taken at three stages of the algorithm:

(a) when all of the eliminations have taken place inside a sector, leaving the interface variables;
(b) at the completion of the elimination phase;
(c) at the completion of the back-substitution.

It was found for this problem that 88% of the total time was taken up in stage (a), that is when each transputer is working independently on its own sector assembling elements, storing and eliminating, 9% in (b), completing the elimination phase and 3% in (c). A second test using 8 processors and 1200 variables gave results of 85%, 11% and 4% respectively. So a processor efficiency of about 90% is achieved for this algorithm using up to eight transputers. Obviously, as the number of variables is increased, with the same number of processors, the efficiency will increase as more work will be performed on the variables inside a sector.

The preliminary results show that we have almost linear speed-up. Running a problem with 600 variables on four transputers takes 13.5 seconds and with 1200 variables on eight transputers, takes 14.5 seconds. There is some degradation due to increased communication but close to linear speed-up is achieved.

The second test problem was a driven flow over a cavity with Reynolds number one. The primitive variable approach is adopted and each 6-noded triangle has fifteen degrees of freedom. This problem is expressed in an iterative form. The preprocessing algorithm determines the mesh given the number of transputers. A set of processes has been written, in OCCAM 2, which are very similar to NAG finite element software subroutines, to perform all of the necessary operations to assemble an element matrix. These processes replace the element matrix process in the first test and then with a small number of changes the algorithm runs as before. Timings have been undertaken as for the first problem and these show even higher efficiencies due to the large amount of processing taking place in the assembly. For example a problem with 600 variables spends 90% of the time in stage (a). Similar results are obtained for eight processors. The times for one iteration using four transputers with 600 variables and eight transputers with 1200 variables are approximately 20.5 and 22 seconds respectively. By way of comparison we have run the same problem on a VAX 8650. We used a modified frontal scheme and for a standard mesh with 1202 variables one iteration takes 31.5 seconds.

Conclusion

This work demonstrates the feasibility of using the multifrontal method as a solver for finite element systems on transputer networks. More tests are required to confirm that the algorithm displays close to linear speedup. The method can be extended to treat non-convex domains by considering them as the union of convex domains and then dividing accordingly. A multifrontal scheme may replace the frontal scheme placed on each transputer. Considerably more work needs to be done on these schemes. However, the work is continuing and will be presented at a later date.

References

1. Hill G. Transputer Networks using the IMSB003 Technical Note 13. INMOS 1987.

2. Irons, B.M. A frontal solution program for finite element analysis. Int. J. Num. Meth. Eng. 2, 5-32. 1970.

3. Hood, P. Frontal solution program for unsymmetric matrices. Int. J. Num. Meth. Eng, 10, 379-399. 1976.

4. Duff, I.S. Design features of a code for solving sparse unsymmetric linear systems out of core. Harwell Report. CSS 89. 1981.

5. Duff, I.S. MA32 - a package for solving sparse unsymmetric systems using the frontal method. Harwell Report. AERE R/10079. 1981.

6. Reid, J.K. TREESOLVE, A FORTRAN package for solving large sets of linear finite-element equations. Harwell Report CSS 155. 1984.

7. Duff, I.S. and Reid J.K. The multifrontal solution of unsymmetric sets of linear equations. Harwell Report. CSS 133. 1983.

8. Duff, I.S. Parallel implementation of multifrontal schemes. Harwell Report. CSS174. 1985.

9. Duff, I.S. Multiprocessing a sparse matrix code on the Alliant FX/8. Harwell Report. CSS 210. 1988.

APPENDIX

II

A paper published in the "International Journal for Numerical Methods in Fluids", discussing further work undertaken on a transputer system for solving fluid mechanical finite element systems.

INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 9, 731–740 (1989)

MULTIFRONTS AND TRANSPUTER NETWORKS FOR SOLVING FLUID MECHANICAL FINITE ELEMENT SYSTEMS

R. G. MILES* AND S. P. HAVARD

Department of Mathematics and Computing, Polytechnic of Wales, Pontypridd, Mid. Glam., U.K.

SUMMARY

This paper is concerned with the implementation of a multifrontal solver on a network of transputers. We briefly discuss the transputer, outline the frontal and multifrontal schemes and consider the implementation of these schemes on the network. Results are presented for two test problems in fluid mechanics showing that the solver displays close to linear speed-up.

KEY WORDS Multifrontal solver Transputer networks Fluid mechanics Finite element method

1. INTRODUCTION

Currently the solution of large fluid mechanics problems requires the use of large, powerful computers which are expensive to use and may involve long 'turn-round' times due to unavailability. A number of multiprocessor computers have been developed. These computers offer comparable speed to large serial computers at a fraction of the cost. Also the use of these computers results in a considerable reduction in total elapsed time for the above problems.

The introduction of the transputer, a basic building block for forming parallel processors, has made possible a powerful workstation hosted by a personal computer. This type of architecture is scalable in the sense that more processors may be added without any degradation in performance. In this paper we will present results using up to 16 transputers; however, we believe that the work will be applicable for a considerably higher number of processors.

To make full use of the available power, attention must be focused on developing new algorithms for these types of architecture. The fundamental characteristics of finite element method (FEM) calculations are that they are computationally intensive and use a lot of storage. They also contain a lot of inherent parallelism.

The purpose of this paper is to consider the feasibility of using a network of transputers to solve some problems in fluid mechanics using the FEM. Here we will restrict ourselves to solving problems with convex domains. The extension of this work to non-convex domains will be the subject of a subsequent paper. In Section 2 the transputer and transputer networks will be briefly described along with the characteristics of these networks which are important for algorithm design.

Within the field of structural mechanics the favoured FEM solution procedure is to partition the finite element mesh into substructures, to eliminate the interior variables and then to solve for

* Present address: The Transputer Centre, Bristol Polytechnic, Bristol BS16 IQY U.K.

0271-2091/89/060731-10$05.00

Received August 1988

the variables on the boundaries between substructures by using a preconditioned conjugate gradient method.[1] The choice of preconditioner is crucial and is the source of a lot of discussion. In this paper we will use the multifrontal method introduced by Duff and Reid[2] and Reid.[3] This method also partitions the finite element mesh and then a frontal method is applied to each substructure to eliminate the interior nodes. This generates a set of substructure matrices which may themselves be considered as superelement matrices. These superelements may again be used with the frontal method to complete the process of elimination. A back-substitution yields the solution. In Section 3 frontal and multifrontal methods will be discussed and in Section 4 we will show how these methods may be implemented on transputer networks. The implementation will be tested on two problems: firstly on Laplace's equation in a square region with Dirichlet boundary conditions and secondly on a driven cavity flow. The results of these tests are given in Section 5. In Section 6 we will conclude by comparing one of the second test results with those from a serial computer and by discussing extensions of this work.

2. TRANSPUTER NETWORKS

The transputer is a single-chip microprocessor which has been designed specifically to facilitate concurrent processing. Each chip contains a processor, memory and communication links. It is by use of these links that connections can be made to other transputers. The T414 has a 32-bit RISC microprocessor, 4 Kbytes of on-chip RAM and four communication links. The chip also has interface circuitry and bus logic to allow for external memory to be attached. On this chip all floating-point operations are performed via software; however, the T800 chip has a 64-bit floating-point hardware unit. The four interface communication links allow for communication between transputers on a point-to-point basis. Since the links and the processor are largely independent, the processor can continue executing other instructions whilst the transputer is passing messages via the links. By passing messages, information may be transferred from one transputer to another, and consequently a network of transputers may be constructed.

The OCCAM programming language was designed prior to the development of the transputer, and transputers implement this language very efficiently. Although other languages, such as FORTRAN and PASCAL, may be used to implement processes on transputers, concurrency is only available through OCCAM. So the creation of parallel processes and the passing of messages must be done through OCCAM, and in this paper all of the algorithms have been implemented in this language.

Various building blocks for constructing transputer networks are now available, the most common being the four-transputer board. The board that we have used has four transputers, each with 1 Mbyte of memory attached, and two of the links on each are wired to adjacent transputers; however, eight links remain unconnected. This network needs to communicate with the outside world and so another transputer is used as a root linking with the host computer. This is a single transputer board with 2 Mbyte of memory attached. The most common host is an IBM-compatible PC, and these boards fit into the expansion slots. So with two extra boards the IBM-compatible PC will support parallel processing (see Figure 1).

More transputers may need to be configured into the network. INMOS and a number of other companies have fabricated boards with many transputers; however, for reasons of general versatility we have used a number of four-transputer boards. The boards are daisy-chained together and by using the unconnected links we may obtain the desired configuration, be it ring, array or modified hypercube. In Figure 2 is shown a simple eight-transputer two-directional ring. It is quite straightforward to write OCCAM processes to route messages in either the clockwise or anticlockwise directions, so that they get to the required processor in the shortest possible time.

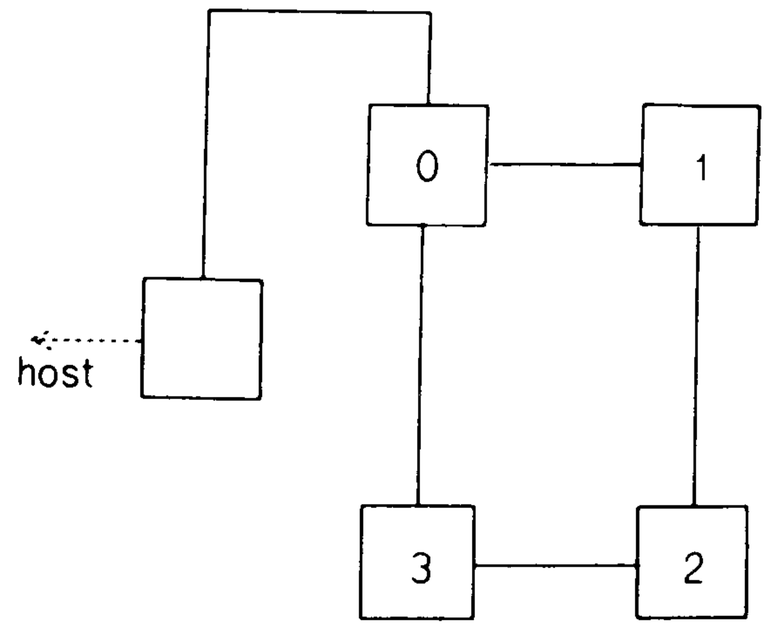


Figure 1. A simple four-transputer network with root to host

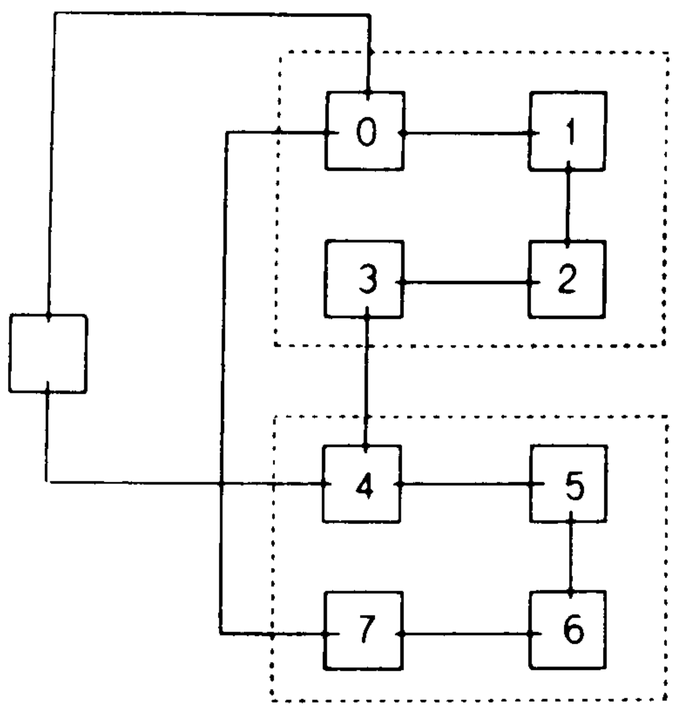


Figure 2. An eight-transputer ring configuration with root

For networks of transputers exceeding eight, a ring configuration is only useful if there is a very small amount of communication or if communication is only with nearest neighbours. For example, with 16 transputers, a ring network has diameter eight, whereas a two-dimensional array has diameter four. There is an INMOS technical note on configuring systems of four-transputer boards.[4]

The results in the last section will be presented for networking four, eight and 16 transputers. In this paper we will develop an algorithm with large granularity so the amount of communication is kept to a minimum. We have performed tests using various configurations, and the time for communication is less than 4% of the total time for even the smallest problem considered in the last section. Consequently the type of configuration is not an issue here. To summarize, the important characteristics of these networks are:

(a) Each transputer has its own memory of 1 Mbyte or more.
(b) Each transputer is controlled by its own set of instructions.
(c) Transputers communicate by passing messages, and it is straightforward to pass messages from one transputer to another in the network.
(d) There may well be a large number of transputers in the network.
(e) There is no shared memory and no shared clock; conseqeuntly each transputer is a totally independent unit which can only communicate with other transputers by message passing.

3. FRONTAL AND MULTIFRONTAL SYSTEMS

The frontal scheme as proposed by Irons[5] is a modified Gauss elimination procedure which is very frequently used with the FEM. The method may be divided into three parts, namely, the assembly of the equations, the elimination and the back-substitution. Irons suggested that elimination should take place whenever possible, rather than assembling the whole system and then beginning the elimination process. The method can be visualized using Figure 3, assuming one degree of freedom at each node. In this figure the element matrices for elements denoted 'A' will have been assembled and contributions made to the system matrix. The contributions to nodes 1, 2, 3, 4, 5 are complete and so the rows associated with these nodes will have been eliminated. Consequently the system matrix will be of order four and will have rows associated with nodes 6, 7, 8, 9. So in this example the front connects nodes 6, 7, 8, 9. As the front progresses across the finite element mesh, elements behind it will have been assembled and contributions made to the system matrix. Those nodes introduced, but not on the front, will have all of their contributions completed and so the rows associated with these nodes will have been eliminated. The elements ahead of the front have yet to be assembled and consequently their nodes are either on the front or have yet to enter the system.

The basic frontal scheme may be summarized as having the following steps:

(a) Assemble the matrix for an element.
(b) Incorporate this into the system matrix.
(c) See if any rows correspond to fully contributed nodes; if so, store the row, the variables associated with it, and the right-hand side, and then eliminate this row.
(d) Continue (a), (b) and (c) until all elements have been assembled, then continue (c) to complete the elimination.
(e) Perform the back-substitution.

In practice a number of improvements have been made to this scheme, the most important of which, for non-positive definite matrices, is to try to retain accuracy by allowing the system matrix to build up until it reaches a level where it will have a number of rows to eliminate. Some of these, enough to allow another element to be assembled, are eliminated using the largest pivots.

This scheme was modified by Hood[6] to treat unsymmetric matrices with symmetric sparsity pattern and by Duff[7,8] who treated the unsymmetric patterns. Pivots are not necessarily diagonal elements and not all fully contributed variables are eliminated. Reid[3] suggested substructuring the finite element problem and used a tree to represent the order of assembly. There may be many

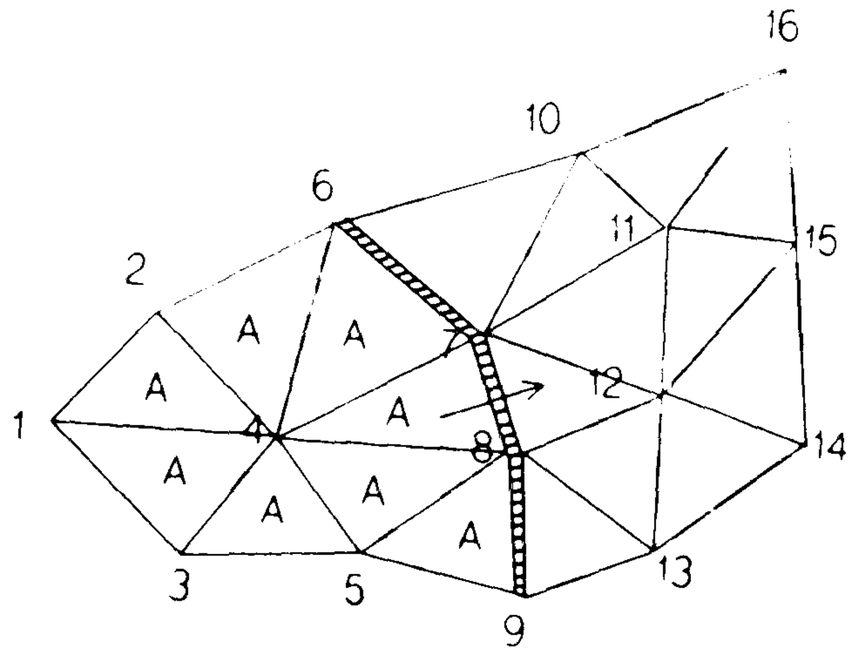


Figure 3. Finite element mesh with one front

frontal matrices and the method is called multifrontal. This may be visualized using Figure 4. Here there are three substructures. The numbers 1, 2, 3 in an element indicate to which substructure it belongs. In substructure 1 the front has assembled three elements and, after performing eliminations, the matrix associated with this front has order three associated with nodes 2, 4, 5. In the other two substructures the fronts have assembled four and seven elements and the frontal matrices are associated with nodes 6, 7, 11, 15 and nodes 9, 8, 7, 11, 15 respectively. In these two substructures the frontal method is complete and these two frontal matrices may be considered as superelement matrices. It may now be decided to merge these two superelement matrices, i.e. to coalesce the two fronts. This process is a small generalization of step (b) above. On performing this, the rows associated with nodes 11 and 15 become fully contributed and then are available for elimination. These two fronts, having been coalesced, have now reduced to the single front connecting nodes 6, 7, 8, 9.

Duff and Reid[2] applied the multifrontal scheme to solve unsymmetric sets of linear equations. They use the flexibility that this introduces to make pivot choices which are economical in arithmetic operations. Duff[9,10] has also considered the application of these schemes for solving unsymmetric sets of linear equations on shared memory multiprocessor systems.

4. MULTIFRONTAL SCHEMES ON TRANSPUTER NETWORKS

When implementing an algorithm on a transputer network, there are three principal considerations that must be taken into account. These are:

(a) granularity
(b) load balancing
(c) an overall organisational structure.

In the FEM a lot of data is manipulated and large matrices are generated. Using relatively small numbers of transputers, most code and data will be located on off-chip memory; consequently we wish to partition the data and replicate the algorithm on each transputer. So an algorithm will be developed with large granularity. The speed at which the scheme works is dependent upon how efficiently each processor is used. If one transputer is given considerably more work to do than the others, and if other processors are waiting for information from this transputer, then they will be idle. So the transputer given the most work may effectively control the network. An algorithm is required which balances the work over the available transputers. As indicated above, the same

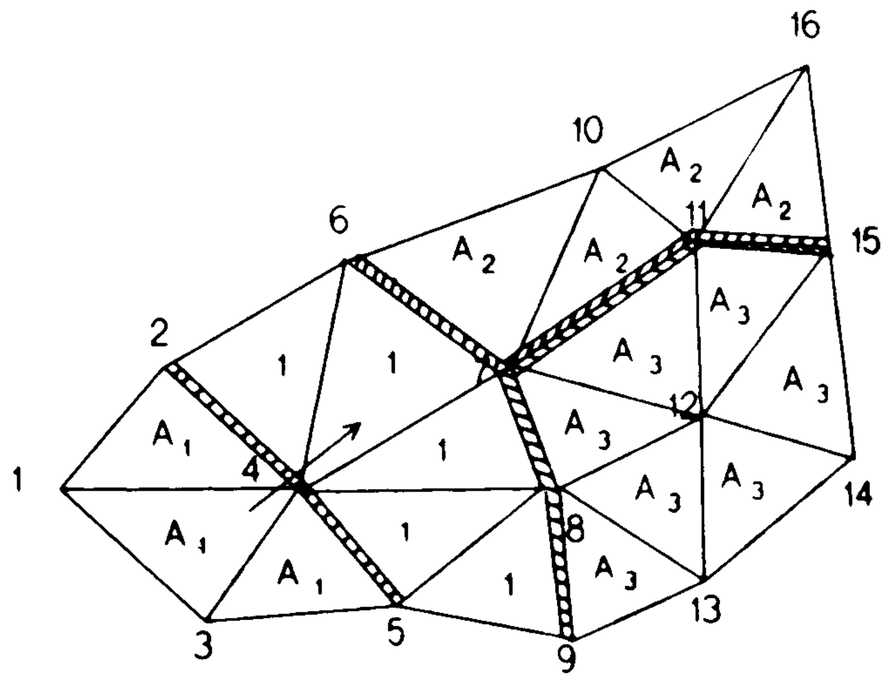


Figure 4. Finite element mesh with three substructures

process is to be allocated to each transputer so a simple data structure is required to organize the passing of messages from one transputer to another. The multifrontal scheme seems very suitable with respect to the above considerations.

A preprocessing algorithm is used which runs on the root transputer to divide the finite element mesh over the number of available transputers and to allow a frontal, or multifrontal, scheme to work on each subdivision. In this study a modified frontal scheme has been placed on each transputer, thus ensuring large granularity. The way the finite element mesh is divided among the transputers is crucial both with respect to load balancing and to the efficiency of the multifrontal method. To balance the load on the network, we want to give each transputer an equivalent amount of work to perform, especially during the first stage of the method when all of the internal variables are being eliminated. Take the example of a square region with n^2 interior nodes, each with one degree of freedom, on a square mesh. If we have eight transputers we cannot easily divide this region to obtain balanced and efficient processing. For example, if the mesh is subdivided by columns then the first and last divisions will have final frontal matrices of order n whereas the ones in between will have order $2n$. Doubling the size of the front crucially affects the efficiency of the method. We require to balance the load and to keep the frontal matrices as small as possible.

We have only worked with convex regions and have decided to use a hub and spokes. An example of this type of mesh for a square region with eight transputers is shown in Figure 5. This type of substructuring ensures a balanced load and minimizes the size of the frontal matrix. The domain is divided into sectors dependent on the number of available transputers, and then the finite element mesh is formed accordingly. Two types of mesh have been used within the sector: a mesh composed entirely of triangles and a mesh composed of quadrilaterals and triangles. In general we prefer the latter type, since this mesh will simplify the generation of contour plots.

Each transputer is sent the information relating to its sector along with information about the global structure of the mesh. The latter is necessary when transputers pass information from one to another.

Each transputer works on its own sector and, when completed, the matrices will be associated only with interface variables (i.e. those on the boundaries between sectors); all other variables will

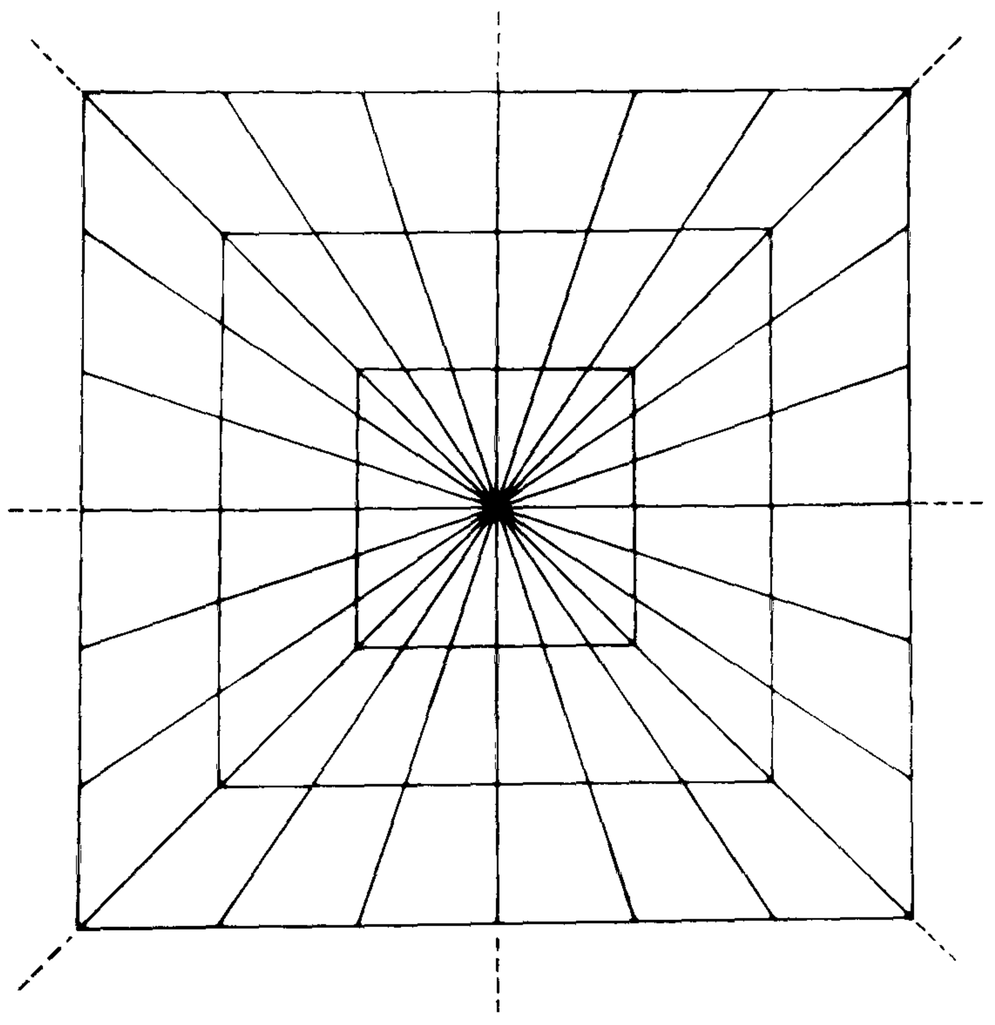


Figure 5. A simple mesh for a square domain with eight transputers

have had their associated rows stored and have been eliminated. Information must then be passed between transputers, and by combining matrices more variables have full contributions and so become available for elimination. The best organizational structure for this is a tree (Figure 6). As the tree is traversed, each transputer knows whether it is to receive or send data and is in a suitable state to do so. For the elimination phase the tree is traversed from the leaves towards the root and then in reverse for the back-substitution.

As the tree is traversed during the elimination phase, the efficiency may be improved by sharing work over the available processors. At level 0 of the tree, each transputer is working independently on its own sector of data. When this has finished, each odd-numbered transputer in Figure 6, T_{2j+1} ($j = 0 \ldots 3$), sends the resulting frontal system matrix to T_{2j}. This local root transputer coalesces the matrices and, if the resulting matrix is sufficiently large, with sufficient rows to be eliminated, it will allocate work to its associated transputer. The associated transputers are shown in dashed line blocks in the figure. The same principle is applied to the other levels of the tree apart from the highest level. In general suppose that we have t transputers. So at level p ($0 < p < \log_2 t$) transputer T_{i2^p} ($i = 0 \ldots t/2^{p-1}$) receives information from $T_{(i2^p + 2^{p-1})}$ and uses some of the associated transputers $T_{(i2^p + j)}$ ($j = 1 \ldots 2^p - 1$) to assist in the elimination.

This sharing of work in the elimination process for higher levels of the tree is best explained at level 1 with, say, transputers T_0 and T_1 working on adjacent sectors of the finite element mesh. When the two transputers have completed the level 0 work, T_1 sends its frontal matrix for sector 1 to T_0. T_0 coalesces the received matrix with its own and this results in a number of variables having full contributions and so being available for elimination. Each of the rows associated with these variables must be eliminated from all of the other rows. At this level two transputers are available, so the local root transputer T_0 forms a packet of all the fully contributed rows and half of the other rows. This it sends to T_1. Now these two transputers can work independently on the elimination. At completion, T_1 returns its half of the matrix to T_0 which together with that on T_0 forms the complete matrix. There is some necessary duplication of work in this process, but a considerable time saving may be achieved.

This same idea may be used at all levels of the tree except the top level. The local root transputer makes the decision of how many assistants to use depending on the size of the matrix and the number of rows to be eliminated. At the top level all rows are now available for elimination, and so very much more communication is required if more than one transputer is used; consequently we have not coded a shared process for this level. We have looked at the time taken in that part of the elimination phase where processor communication is involved, i.e. at

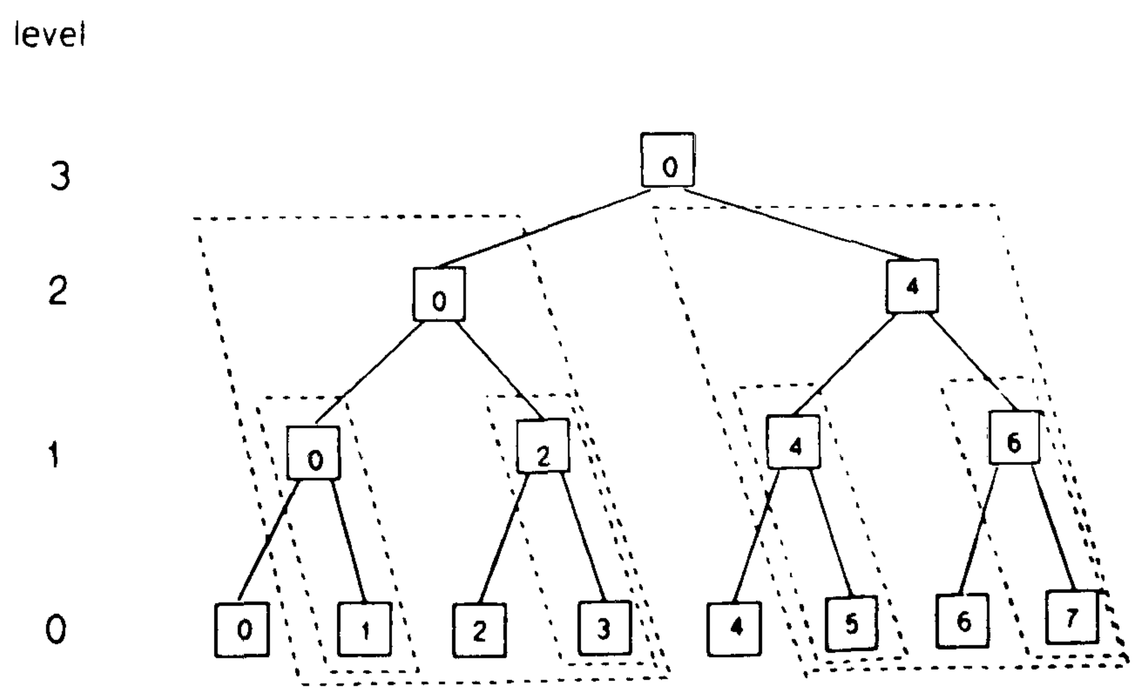


Figure 6. Tree stucture for eight transputers

levels exceeding 0. We have found that up to a 50% saving in time is achieved by using the above approach rather than having no sharing of resources. At each level the number of eliminated rows is stored for use in the back-substitution.

At each level of the back-substitution, the transputer knows how many values are to be determined. These values are then passed to the next transputer so that it can continue the process. During this part of the algorithm some transputers are idle, waiting for values to be passed to them from higher levels; however, at the lowest level all are active, working on their own sector.

5. RESULTS

The first test problem for this algorithm was Laplace's equation in a square region with Dirichlet boundary conditions. We have used a triangular mesh with one degree of freedom per node; consequently the element matrix is extremely easy to form.

First of all the efficiency was examined. To assess this it is useful to divide the algorithm into three stages:

(a) the level 0 eliminations in Figure 6
(b) the other levels of the elimination phase in Figure 6
(c) the back-substitution.

For a number of problems separate timings have been taken in each of these three stages.

The results are presented in Figure 7 for 597, 1193 and 2385 variables using four, eight and 16 transputers respectively. For simplicity we will use the pair (597, 4) to denote a problem with 597

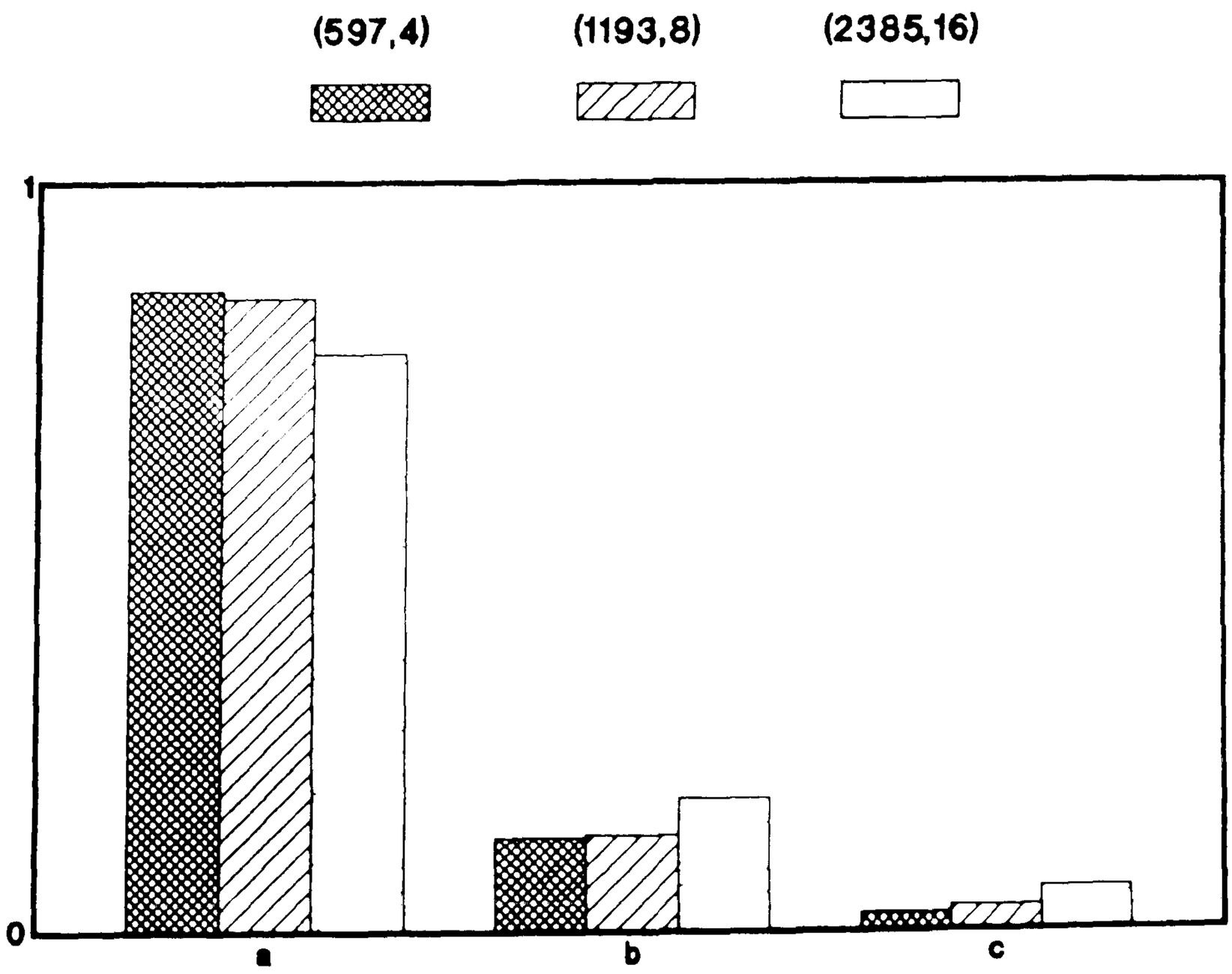


Figure 7. Histograms showing the percentage of time spent in stages (a), (b) and (c)

variables and four transputers. For the stage (a) eliminations the transputers are working entirely independently and during stage (b) sharing of resources may be taking place, so we can estimate that an efficiency greater than 85% is achieved in all cases. The results show almost linear speed-up. Running the (597, 4), (1193, 8) and (2385, 16) problems takes approximately 13·4, 14·6 and 16 s respectively. When the number of transputers is increased, more levels are introduced into the organizational tree. Consequently a smaller percentage of the time is spent in stage (a) and larger percentages in (b) and (c); however, close to linear speed-up is achieved. We have also solved some large problems; for example, the (5745, 16) problem takes approximately 58 s.

The second test problem was a driven flow over a cavity with Reynolds number one. The primitive variable approach is adopted and the non-linearity due to the advection terms of the momentum equations is handled by expressing the problem in iterative form. The preprocessing algorithm determines the mesh given the number of transputers. We have used six-noded triangles with 15 degrees of freedom and eight-noded quadrilaterals with 20 degrees of freedom, so that the pressure shape functions are one order lower than those for velocity. A set of processes has been written, in OCCAM, which are very similar to the NAG finite element software subroutines, to perform all of the necessary operations to assemble an element matrix. In particular the code must first of all determine the type of element so that the correct basis functions and integration routines are selected. It must also assign a suitable numbering system to the variables. These processes replace the element matrix process in the first test and then with a small number of changes the code runs as before. These changes are due principally to there being more than one degree of freedom at each node, and to this degree of freedom being three at some nodes and two at others.

Timings have been taken as for the first test problems and results very similar to those in Figure 7 are achieved. The stage (a) percentages are slightly higher owing to the large amount of processing taking place in the assembly, and consequently slightly higher transputer efficiency is achieved. The times for one iteration of the (591, 4), (1179, 8) and (2355, 16) problems were approximately 20·6, 22 and 23·7 s respectively. Again there is slight degradation when the number of transputers is increased, but close to linear speed-up is achieved. We have also run one larger problem, (5809, 16), and this takes approximately 2 min to complete one iteration.

6. CONCLUSIONS

This paper demonstrates the feasibility of using the multifrontal method as solver for finite element systems, with convex domains, on transputer networks. It has been shown that high transputer efficiency is achieved and that the algorithm displays close to linear speed-up. By way of comparison we have run the second test problem on a VAX 86 50. We used a modified frontal scheme and a mesh with 1202 variables. One iteration on the VAX took 31·5 CPU seconds. This compares with the (1179, 8) problem which took 22 s. On reading the literature for the transputer, it may be considered that the above result is slower than expected; however, it must be remembered that we have nearly all code and data residing on off-chip memory. Hockney and Jesshope[11] have performed some benchmark tests to check the difference in using on-chip and off-chip memory and they report a 3:1 time difference. This, as they point out, is the expected result since, at best, the external memory interface cycles at three transputer clock cycles whereas on-chip memory cycles within a single transputer clock cycle.

It was stated in the Introduction that the work will be applicable to numbers of transputers exceeding 16. This we believe to be the case provided shared processing is used in stage (b) (the stages are indicated in the last section). Then, although a smaller percentage of the time will be

taking place in stage (a), a larger percentage will be taking place in (b) and this will be shared across the processors.

The generalization of this work to non-convex domains is possible by considering them as a union of convex domains and then forming a mesh in each convex region. Sectors which are at the intersection of two regions must be treated carefully since the number of variables and consequently the size of the frontal matrix is considerably bigger. These sectors must have fewer internal variables so as to preserve load balancing. The extension of this work to non-convex regions and consideration of the above issues are to be the subject of further study.

REFERENCES

1. B. Nour-Omid, A. Raefsky and G. Lyzenga, 'Solving finite element equations on concurrent computers', *Parallel Computations and their Impact on Mechanics*, AMD, 86, 1987, pp. 209–227.
2. I. S. Duff and J. K. Reid, 'The multifrontal solution of unsymmetric sets of linear equations', *Harwell Report CSS 133*, 1983.
3. J. K. Reid, 'TREESOLVE, A FORTRAN package for solving large sets of linear finite-element equations', *Harwell Report CSS 155*, 1984.
4. G. Hill, 'Transputer networks using the IMSB003', *Technical Note 13*, INMOS, 1987.
5. B. M. Irons, 'A frontal solution program for finite element analysis', *Int. j. numer. methods eng.*, **4**, 5–32 (1970).
6. P. Hood, 'Frontal solution program for unsymmetric matrices', *Int. j. numer. methods eng.*, **10**, 379–399 (1976).
7. I. S. Duff, 'Design features of a code for solving sparse unsymmetric linear systems out of core', *Harwell Report CSS 89*, 1981.
8. I. S. Duff, 'MA32- a package for solving sparse unsymmetric systems using the frontal method', *Harwell Report AERE R/10079*, 1981.
9. I. S. Duff, 'Parallel implementation of multifrontal schemes', *Harwell Report CSS 174*, 1985.
10. I. S. Duff, 'Multiprocessing a sparse matrix code on the Alliant FX/8', *Harwell Report CSS 210*, 1988.
11. R. W. Hockney and C. R. Jesshope, *Parallel Computers 2*, Adam Hilger, 1988.

APPENDIX

III

A paper presented at the sixth Interantional conference on numerical methods in laminar and turbulent flow, discussing the further development of the multifrontal scheme for MIMD networks and published in the proceedings of the conference.

The Development of a Multifrontal Scheme for MIMD networks

R.G.Miles† and S.P. Havard
Dept. of Mathematics and Computing,Polytechnic of Wales
Mid-Glamorgan,U.K.

Summary

The frontal method is frequently used as a solver for finite element systems.The purpose of this paper is to decribe the implementation of a multifrontal scheme on a MIMD network.The development of the method will be described via the use of three test problems in fluid dynamics.

1. Introduction

The introduction of commercial parallel processors has led to considerable attention being focussed on algorithms suitable for parallel processing. Several workers have developed schemes for the finite element method especially in the area of structural analysis. Nour-Omid et. al. [1] partition the finite element mesh into substructures,eliminate the interior variables and solve for the variables on the boundaries between substructures using a preconditioned conjugate gradient approach.Farhat and Wilson [2] subdivide the mesh and employ an iterative multicolouring scheme.

The aim of this paper is to describe the development of a multifrontal algorithm for solving the Navier-stokes equations on multiprocessor networks. In [3] we used the multifrontal scheme as a method for solving primitive variable systems on MIMD networks. The method is extended in this paper.

† Now at Bristol Polytechnic,Bristol,U.K.

The type of network that we are considering can be characterised as follows:

(i) each processor has its own memory of 1MByte or more;
(ii) each processor is controlled by its own set of instructions;
(iii) processors communicate by passing messages;
(iv) the network may contain many nodes;
(v) there is no shared memory and no shared clock.

Examples of this type of network are the iPSC systems and transputer based systems.

To develop algorithms which work efficiently on these networks we must pay attention to the granularity of the algorithm,the balancing of the load over the available processors and a simple organisational structure.The granularity of the algorithm determines the amount of communication. A large grained approach reduces communication but can result in load balancing problems.The network will only work as fast as the slowest processor. If more work is given to one processor then the rest of the network will be idle waiting for this one to complete,so the work must be balanced across the network.There may be many processors in the network consequently a simple organisational structure is required so that at any stage a processor will know where to send information.

This paper is divided into three sections.In section 2 the frontal and multifrontal schemes are described. The implementation of the multifrontal scheme on networks is discussed in section 3 focussing on the three points made in the last paragraph. In section 4 results will be presented for three test problems namely a driven flow over a square cavity,axisymmetric mixing using a disc rotor and the flow over a backward facing step.

2. Frontal and multifrontal schemes

The schemes have been described in [3],however for coherence they will be described here.It is based on the Frontal Method proposed by Irons[4].This is a modified Gauss Elimination procedure specifically applied to Finite Element formulations.The method is summarised diagrammatically in Fig.1 assuming one degree of freedom at each node.At this point of the process eight elements have been assembled,however the contributions to nodes 1-5 are

complete so the rows associated with these nodes in the system matrix are also complete.Rather than retaining these rows,they are stored and then eliminated from the matrix. The front connects nodes 6-9;these are the active nodes.The nodes behind the front have been

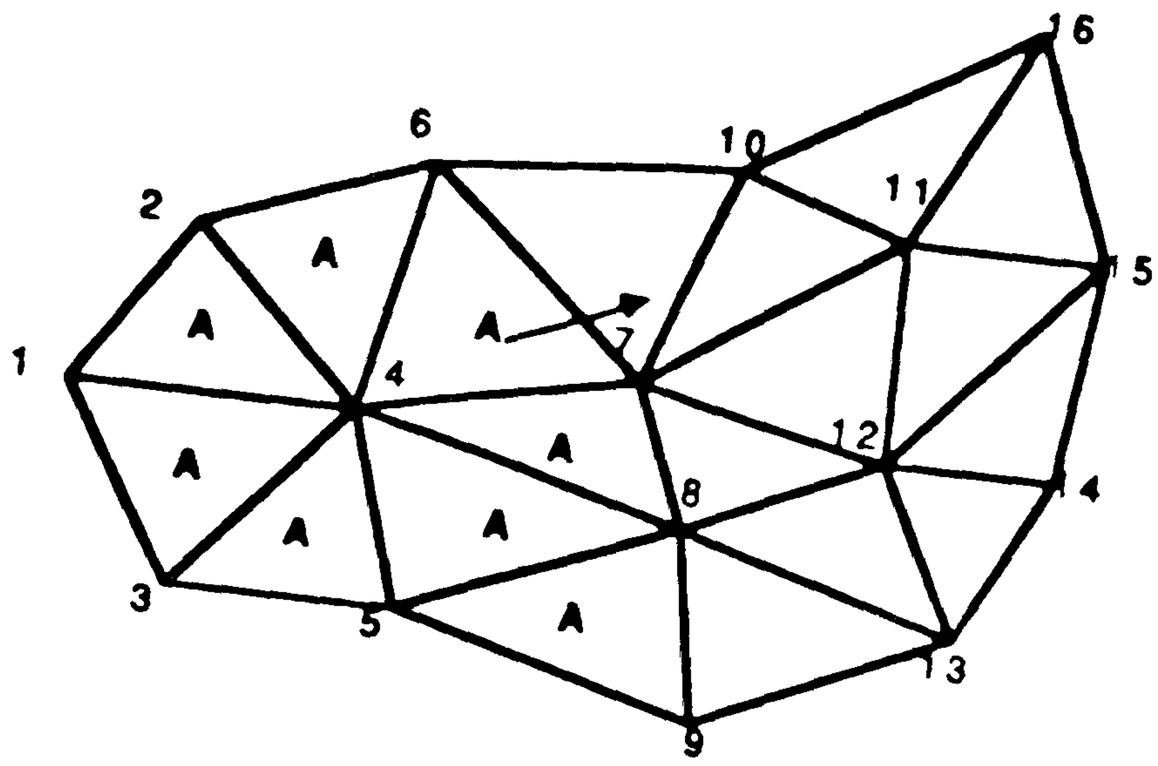


Fig. 1

eliminated whilst those ahead have yet to enter the system.A number of improvements have been suggested to this scheme. Reid[5] suggested substructuring the finite element mesh and used a tree to represent the order of assembly.The method now involves many fronts moving across the domain. This may be visualised in Fig.2.The domain has been sub-structured into three domains and has three fronts operating. In subdomains 2 and 3 all elements have been assembled and eliminations completed so the resulting matrices may be considered as superelement matrices. It may now be decided to merge these two matrices.This corresponds to coalescing the two

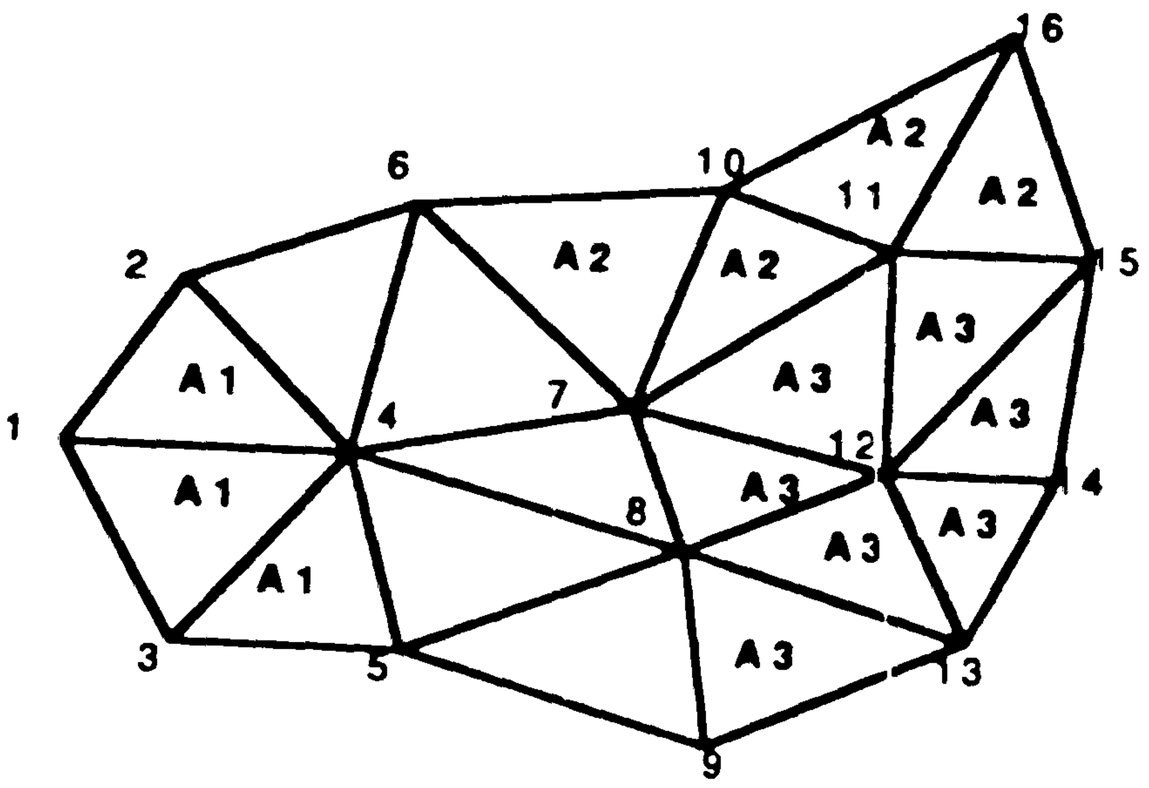


Fig. 2

fronts,consequently nodes which are on both fronts can become available for elimination.The superelement matrices are treated as ordinary element matrices.

3. Implementation of the Multifrontal scheme on MIMD networks

It was stated in the introduction that when implementing an algorithm on a network we must consider granularity,load balancing and an organisational structure for communication. We will partition the domain and place part of it on each processor. A frontal solver is then applied in each partition thus we have a large grain algorithm. To explain the development of our ideas concerning load balancing and organisational structure it is best to take three different types of problem:

(i) problems with a convex domain
  e.g. a driven flow over a square cavity;
(ii) problems with a non-convex domain
  e.g. axisymmetric mixing with a disc rotor
(iii) problems which require a subdomain on a processor
  e.g. a flow over a backward-facing step.

(i) Problems with a convex domain

To balance the load over the available processors we require that each processor has approximately the same amount of work.For a convex domain we have found that to ensure a balanced front width we should use a hub and spokes.Consequently we substructure the domain and form a mesh within each partition. Fig.3 shows a simple partition of a square domain,for eight processors,and the resulting mesh generated in each sector.This domain partitioning ensures a balanced load and minimises the size of the frontal matrix throughout the process.

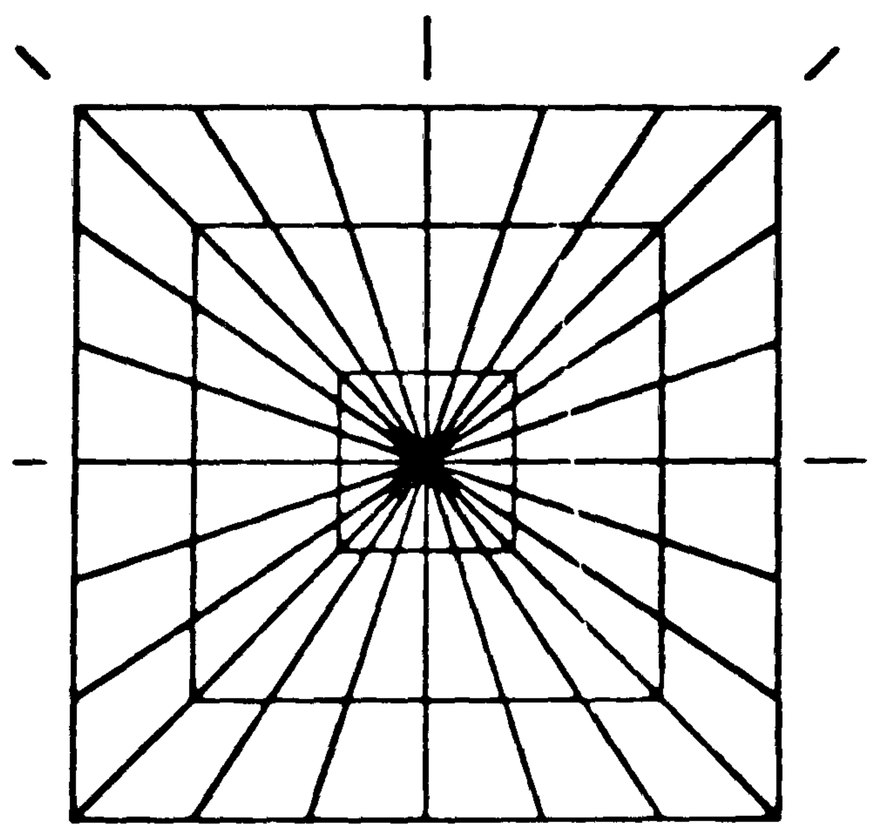


Fig. 3

Associated with this partitioning is an organisational tree (see Fig.4). Each processor will have knowledge of the tree structure and will know whether to send or to receive information and

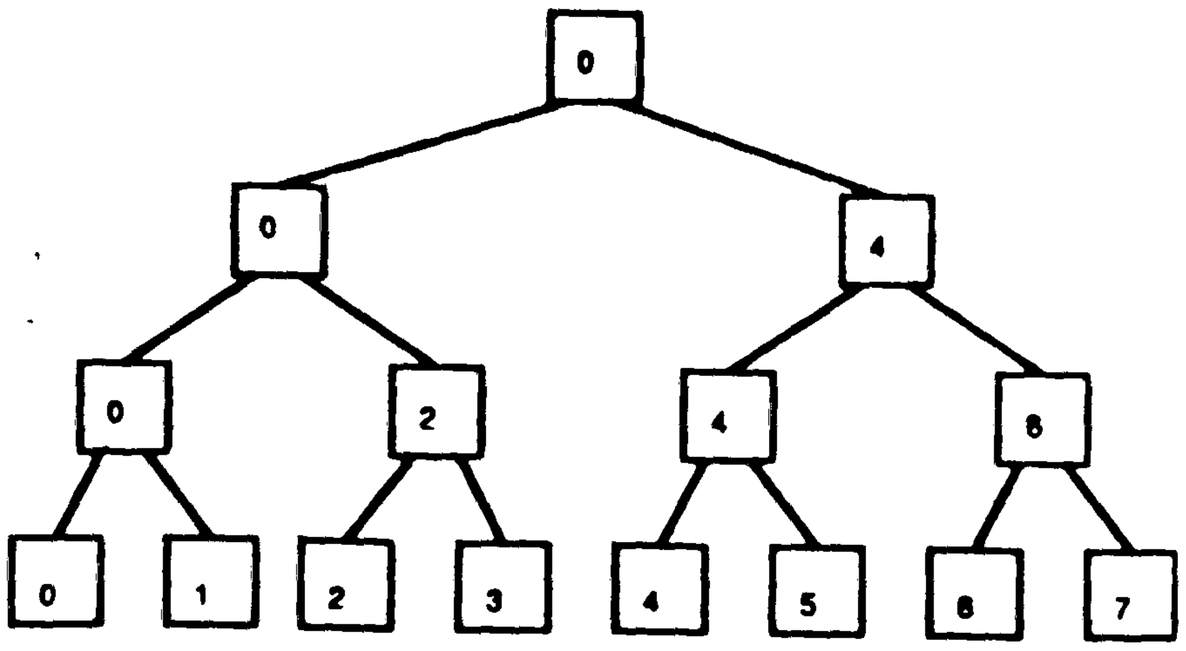


Fig. 4

will be in a suitable state to do so.The tree is traversed from leaves to root during the elimination phase and in reverse for back-substitution.The numbers in the boxes indicate which processors receive information and from whom.It appears that at levels exceeding 0 some processors will become idle.In [3] a refinement is given so that processors becoming idle act as slaves. This improves the efficiency of the elimination at these levels.

(ii) Problems with a non-convex domain

In general non-convex domains will be divided into a union of convex sub-domains and then the procedure outlined above used in each subdomain.As a example consider the domain for axisymmetric mixing using a disc rotor as shown in Fig.5.This domain is subdivided into three convex subdomains and a hub and spokes is used in each. In this case,as we have defined the partitioning, there are two interdomain boundaries and seven intersector boundaries. There are eight sectors and so the organisational tree is as in Fig.4.To achieve load balancing the factors affecting the amount of work in a sector must be identified.

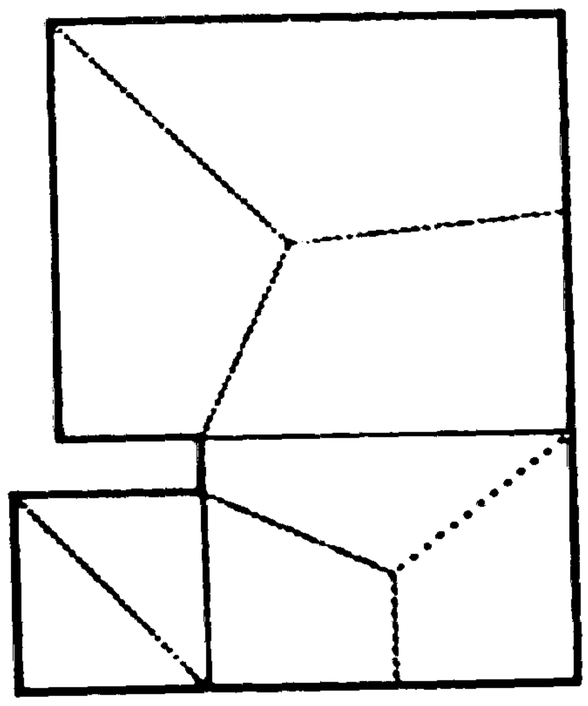


Fig. 5

The three principal factors are the front width,the number of variables and the number of elements.There are two parameters that can be varied,the number of rows and the number of spokes.For problems of significant size the time for manipulation of the system matrix dominates the time spent in element assembly and so the latter will be ignored. To achieve balancing in the elimination at levels exceeding 0 requires that the number of rows is fixed in each sector.So only the number of spokes will be varied to achieve a balanced load.In the case of Fig.5 there are just two different types of sector:

- 2 segments in the domain, no interdomain boundaries;
- 2 segments in the domain,1 interdomain boundary.

However for a different type of partitioning there will be at least two more cases e.g. just one segment with or without an interdomain boundary. An estimate of the load in a sector is expressed as the initial front width multiplied by the number of variables. Both ofthese can be expressed in terms of the number of rows and of spokes. So by evaluating this load factor in each sector we can balance the load across the network once we have chosen the number of rows and of spokes in the most constrained sector.It may be decided that the balanced mesh gives an unacceptable covering in some areas.If this is so then another partitioning of the domain can be adopted (for example moving the sub-domain centres towards an area requiring greater definition).

(iii) Problems which require a sub-domain on a processor

For a problem such as the flow over a backward facing step using a small number of processors it is desirable to place more than one sector on a processor. Consider Fig.6 with four processors and a partitioning as shown. It is desirable to put the whole sub-domain around the inflow onto one processor.

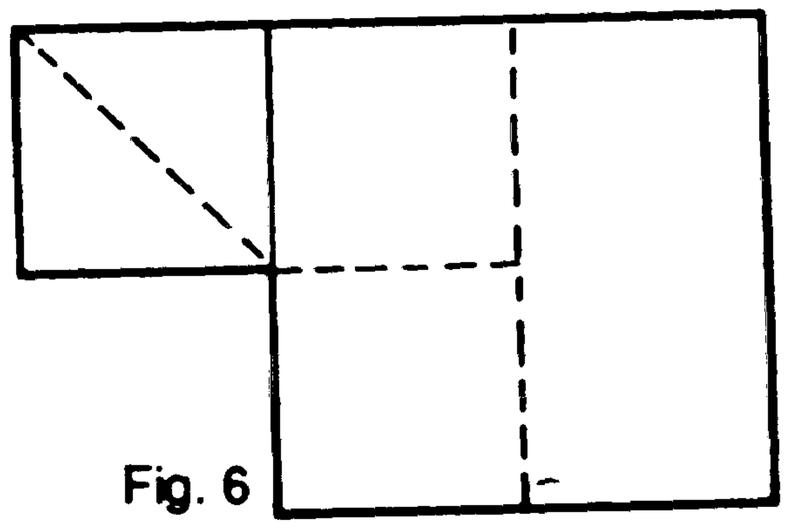


Fig. 6

This can be achieved by modifying the organisational tree as shown in Fig.7.In this instance communication in the area designated — is physical communication between processors whilst that in the

area ---- is logical on one processor. Again a load balancing strategy is used as in (ii).In this case the load factor for each subsector is evaluated and then these are added to determine the total load in the sector. This is then balanced with the loads on the other processors.

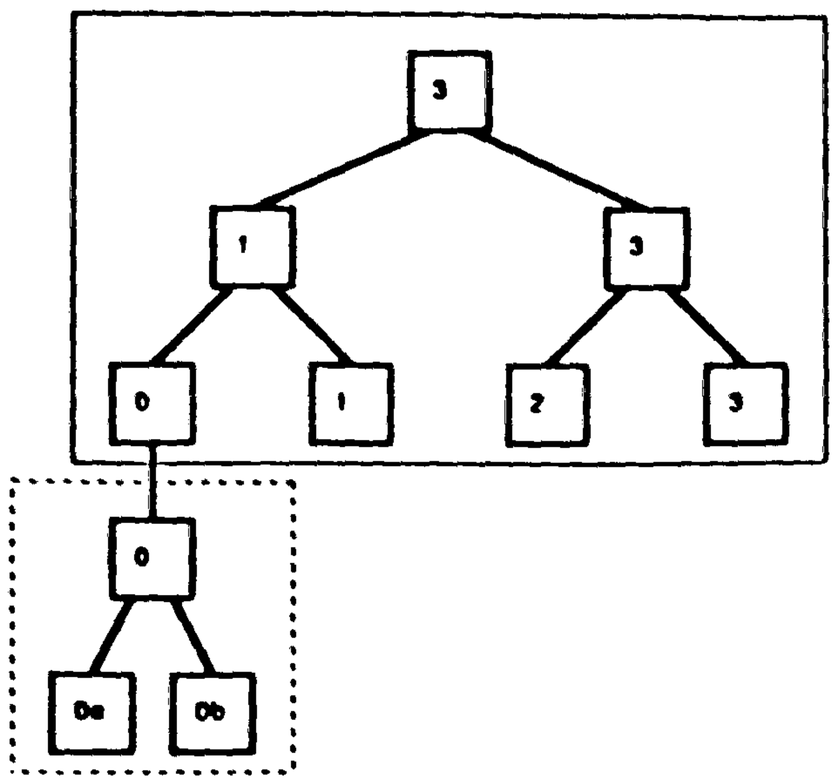


Fig. 7

4. Results

The work described in the last two sections has been implemented on a network of transputers consisting of a network of T800 transputers, each with 1MByte of memory, with a T414 transputer as the root linking with an IBM PC host (Fig. 8).Also attached to the root is a graphics board and screen.The equations are non-linear due to the advection terms and consequently the problem is expressed in iterative form.The root maintains overall control of the solution process.It receives information from each network transputer concerning the maximum change between iterations for the velocity variables in that sector.

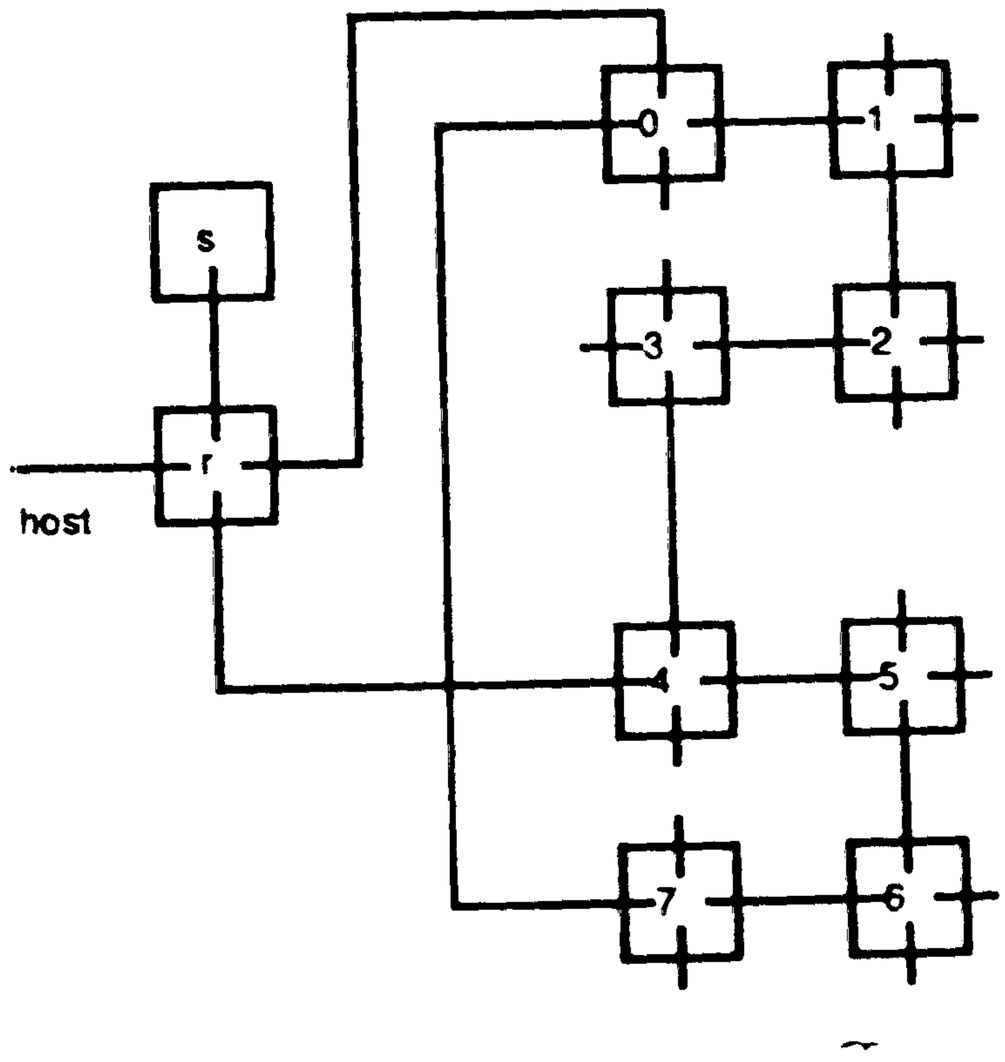


Fig. 8

It can then assess if all of the values are within the specified tolerance and ,if not, sends a message telling each transputer to perform another iteration.Once convergence is achieved the root informs the network that this is so and each processor then determines the stream values,using the same algorithm but with a new element matrix.It then ,for specified contour levels, evaluates the colour value for each screen pixel within the sector. These values are compressed and sent back to the root which sends them to the graphics board and they are displayed on the screen,sector by sector.

Results will now be discussed for each of the problems presented in Section 3 as examples of the three problem types considered:

(i) Driven flow over a square cavity at Re=100

For this problem the elements are either six-noded triangles with 15 degrees of freedom or eight-noded quadrilaterals with 20 degrees of freedom. A set of processes has been written which performs all of the basic operations necessary to assemble the element matrices.To assess the efficiency of the algorithm it is divided into three stages:

a. the level 0 eliminations in Fig.6;
b. the other levels of elimination in Fig.6;
c. the back-substitution.

One iteration has been performed and timings have been taken in these three stages. The results are presented in Fig.9 for 551,1099,2195 variables using 4,8,16 processors.In the figure the pair [551,4] denotes 551 variables and 4 processors.From these results we estimate that a processor efficiency in excess of 85% is achieved. Furthermore the times for one iteration for [551,4],[1099,8] and [2195,16] are 9.3,10.5 and 11.8 seconds respectively,showing that we have close to linear speedup.Contours

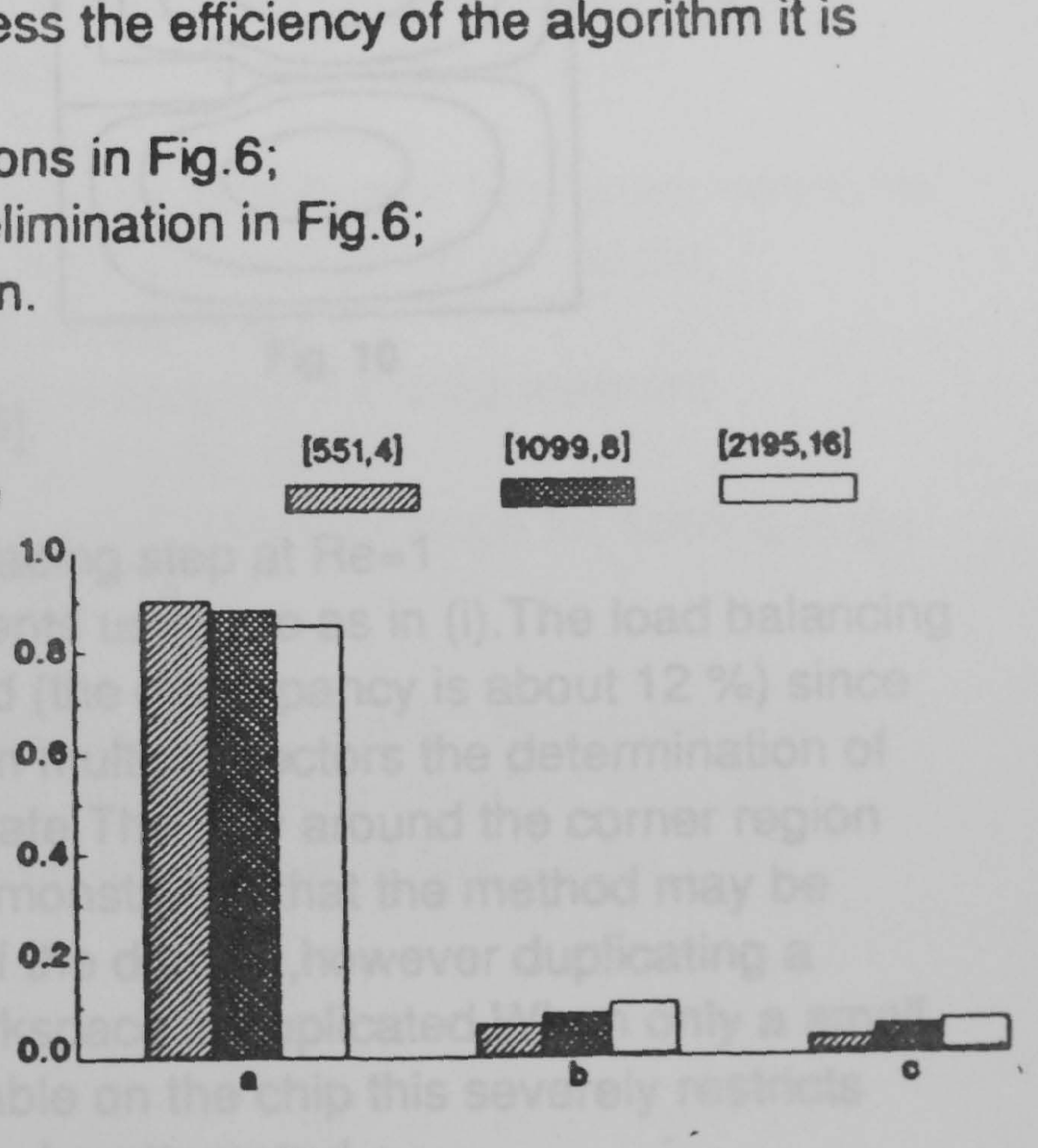


Fig. 9

for the stream function were displayed on the screen and these agree exactly with well known results.

(ii) Axisymmetric mixing using a disc rotor at Re=25

Here the elements are either triangles with 21 degrees of freedom or quadrilaterals with 28 degrees of freedom.The load balancing strategy outlined in Section 3 is applied and for a very coarse mesh we recorded a 15% discrepancy between the first and last transputer completing stage (a), however for finer meshes we recorded discrepancies of less than 7%.This variation is due to neglecting the element assembly time as a parameter in the load balancing.For fine meshes the front width is large and so this dominates the load factor.We have solved the problem using 4391 variables with eight transputers.We recorded a discrepancy of under 4% and one iteration was completed in 121 seconds. Processor efficiency was very high, being in excess of 85%.A copy of the output displayed on the screen is shown in Fig.10. These contours agree with the results of Spragg et.al.[6].

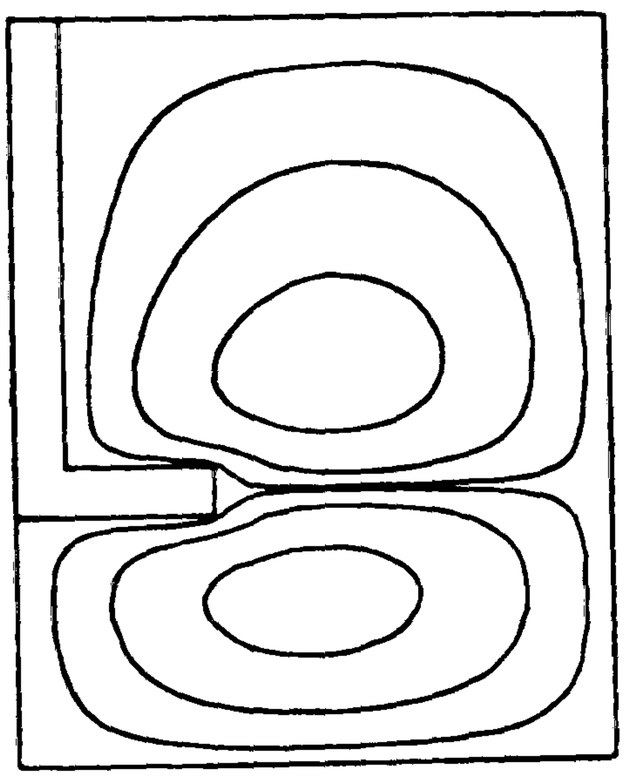


Fig. 10

(iii) Flow over a backward facing step at Re=1

For this problem the elements used are as in (i).The load balancing strategy is not quite so good (the discrepancy is about 12 %) since for the processor working on multiple sectors the determination of the load factor is less accurate.The flow around the corner region is as expected.We have demonstrated that the method may be extended to take account of the domain,however duplicating a sector requires that the workspace is duplicated.When only a small amount of memory is available on the chip this severely restricts the size of problem that may be attempted.

Conclusions

The multifrontal method has been implemented on a network of transputers.The results demonstrate that the method utilises the processors efficiently and displays good speedup.The load balancing strategy applied to non-convex domains gives relatively even balancing across the processors.This method seems quite promising although more work is required to test the algorithm and to optimise the code.

References

1. B. Nour-Omid,A. Raefsky and G. Lyzenga, "Solving finite element equations on concurrent computers",Parallel computations and their impact on Mechanics, AMD 86,209-227,1987
2. C. Farhat and E. Wilson, "A new finite element concurrent computer program architecture",Int.J.Num.Meth.Eng.,24, 1772-1792, 1987
3. R.G. Miles and S.P. Havard,"Multifronts and transputer networks for solving fluid mechanical finite element systems", to be published in Int.J.Num.Meth.Fluids 1989
4. B.M. Irons,"A frontal solution program for finite element analysis",Int.J.Num.Meth.Eng.,4,5-52,1970
5. J.K. Reid,"TREESOLVE,A FORTRAN package for solving large sets of linear finite element equations", Harwell Rpt CSS 155,1984
6. A.J.P. Spragg,S.J.Maskell and M.A. Patrick," Non-Newtonian flow and mixing induced by a spinning disc impeller",Proc.4th Int.Conf.Num.Meth. in laminar and turbulent flow,39-50,1985