A THESIS

entitled

Free Surface Flow and Application to the Filling and Solidification of Liquid Metals into Vessels of Arbitrary Shape

by

(Andrew) Koon Sang CHAN
BSc, MSc, GIMA

Thesis submitted in partial fulfilment of the requirements for the award of the

DEGREE OF DOCTOR OF PHILOSOPHY

of the

UNIVERSITY OF GREENWICH

This research programme was funded by SERC and BNF-Fulmer

Centre for Numerical Modelling and Process Analysis
School of Mathematics, Statistics and Computing
Faculty of Technology
University of Greenwich
London
United Kingdom

February 1994

ABSTRACT

The research work presented herein addresses the problem of the mathematical modelling of the mould filling processes as encountered in the foundry industry.

The quality of castings, especially aerospace components, is primarily pre-determined at the stage of mould filling within the entire casting process. The entrapment of oxide films, air voids and other impurities into the cast, caused by waves and the breaking of the molten metal surface during filling must be avoided. Otherwise, substandard casting products will result which cost the foundry industry millions of pounds in lost revenue.

A three-dimensional control-volume, free surface flow technique known as the *Scalar Equation Algorithm (SEA)* has been developed as an attachment to the PHOENICS and Harwell-FLOW3D CFD codes for this study. The SEA technique uses a conserved scalar variable to represent the liquid, with an adaptation of the van Leer TVD scheme to define the instantaneous position of the interface. It is similar to the approach used by the well known Volume Of Fluid (VOF) method. However, the SEA technique deals with both air and liquid explicitly, whereas the VOF method does not.

A technique has also been developed to allow the liquid temperature to be determined from a conserved 'mixture' enthalpy. The liquid temperature is subsequently used in a solidification algorithm to simulate the effect of phase change.

The filling model without heat transfer and solidification has been validated against experimental data in both water experiments and actual mould filling experiments. The capability of the SEA method in capturing convoluted waves and air voids has been successfully demonstrated in an example of filling part of a mould running system.
It has also been compared against the predictions from the SOLution Algorithm-Volume Of Fluid (SOLA-VOF) and Marker And Cell (MAC) methods. Examples of the developed filling model coupled with heat transfer and solidification are also given.

To increase computational speed, the filling model has been implemented into a parallelised version of the Harwell-FLOW3D CFD code. A speed-up of up to 80% has been achieved by using a network of heterogeneous processing nodes. Each node consists of an Intel i860 vector processor and an Inmos T800 transputer.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisors Dr. Koulis Pericleous and Professor Mark Cross for their invaluable advice, skilful guidance and continual encouragement received during the course of this research. Without their support and patience, this research would not have been possible.

My sincere gratitude is also extended to Drs. Peter Chow, Nicole Hoffmann, Steve Johnson, Mayur Patel, Chris Bailey, Professor Edwin Galea, Mr. Keith Preddy and Dr. Brian Denton (of BNF-Fulmer Metals Technology) for the invaluable discussions, assistance and encouragement at various stages of the research.

I would like to dedicate this thesis to my family and especially to my wife, Lisa, who has supported me unconditionally for the past four years and suffered a lot of hardship as a result.

I also wish to thank all my friends and colleagues at the School of Mathematics, Statistics and Computing for their advice and assistance. I would like to thank, in particular Dr. Yvonne Fryer, Messrs. M Afzal, Peter Lawrence, Miltos Petridis and Mick Hughes at the Centre for Numerical Modelling and Process Analysis, for they have contributed stimulating discussions, various ideas and more importantly a friendly atmosphere to work in. My gratitude is also extended to the staff of the Computing Centre and the Faculty's computing support team, in particular Messrs. Robert Paynter, Ian Lee, John Watts and Sean O'Sullivan for providing me with the vital computer resources. The invaluable help provided by Mr. Alan Leung in the preparation of some of the diagrams is greatly appreciated.

The financial supports provided by the Science and Engineering Research Centre and BNF-Fulmer Metals Technology are gratefully acknowledged. Finally, I would like to thank CHAM Limited and AEA Technology for their kind permissions to use the PHOENICS and FLOW3D codes.

CONTENTS

NOMENCLATURE

English Characters

A	Surface area of a cell-face (m^2)
C	Convective mass flux (Chp. 2)
C	A large positive constant (Chp. 6)
C	Variable for Concentration (Chp. 2)
C_p	Specific heat capacity (J/kg/°K)
D	Diffusive mass flux
E	Cell centre of the control-volume at the *east side* of cell P
E_p	Computational efficiency of p number of processors
F	Volume fraction used in the VOF method
$\underline{F}$	Velocity vector field defined on a volume V
H	Cell centre of the control-volume at the *high side* of cell P
H	Height of the free surface (Chp. 1) (m)
K	Permeability of porous materials
L	Cell centre of the control-volume at the *low side* of cell P
L	Latent heat energy of phase change (Chp. 6) (J/kg)
M_s	Volumetric specific surface area of a dendrite arm
N	Cell centre of the control-volume at the *north side* of cell P
P	Cell centre of a control-volume
R	Generic variable
S	Source or sink term
S	Cell centre of the control-volume at the *south side* of cell P
S	Closed surface around the volume V (Chp. 2) (m^2)
S_C	Constant part of a linearised source term
S_P	Coefficient of the variable part of a linearised source term
S_B	Source term for buoyancy
S_{ext}	External source or sink term
S_h	Source term for the enthalpy equation
$\overline{S}_h$	Source term for the 'mixture' enthalpy
S_p	Speed-up factor for p number of processors running in parallel
S_u	Darcy term for the u-momentum equation
S_v	Darcy term for the v-momentum equation
S_w	Darcy term for the w-momentum equation
S_Φ	Source or sink term for a generic variable, Φ
T	Temperature (°K)
$\mathcal{T}$	Dominant temperature or averaged temperature (°K)
T_l	Liquidus Temperature (°K)
T_{ref}	Reference Temperature (°K)
T_s	Solidus Temperature (°K)
V	Volume of a control volume cell (m^3)
W	Cell centre of the control-volume at the *west side* of cell P

a	Variable for coefficients in all discretised equations (Chp. 2)
b	Variable for all other terms which do not belong to the transient, convective or diffusive term of a transport equation (Chp. 2)
e	Position at the mid point of the cell-face between cells P and E (where velocity u is stored)
f_l	Liquid fraction for solidification (Chp. 6)
g	Gravitational force (usually taken to be 9.81 m/s^2)
h	Position at the mid point of the cell-face between cells P and H (where velocity w is stored)
h	Variable for enthalpy (Chp. 2) (J/kg)
$\boldsymbol{h}$	'Mixture' enthalpy (Chp. 6) (J/kg)
$\bar{h}$	Mass averaged enthalpy (J/kg)
k	Thermal conductivity (W/m/°K)
l	Position at the mid point of the cell-face between cells P and L (where velocity w is stored)
ln	Natural logarithm (Chp. 2)
max	Maximum value
min	Minimum value
n	Position at the mid point of the cell-face between cells P and N (where velocity v is stored)
$\underline{n}$	Unit outer normal on surface S
p	Variable for Pressure (N/m^2)
r	Blockage factor (Chp. 1)
s	Position at the mid point of the cell-face between cells P and S (where velocity v is stored)
sgn	The sign of a variable
t	Time (s)
t_p	Execution time of p processors (s)
t_s	Execution time of a single processor (s)
u	Velocity component along the x-axis (m/s)
u_p	Velocity of a particle travels along the x-axis (m/s)
$\underline{u}$	Velocity vector (m/s)
v	Velocity component along the y-axis (m/s)
v_p	Velocity of a particle travels along the y-axis (m/s)
w	Position at the mid point of the cell-face between cells P and W (where velocity u is stored)
w	Velocity component along the z-axis (m/s)
x_p	Location of the particle along the x-axis in cartesian coordinate
y_p	Location of the particle along the y-axis in cartesian coordinate

Greek Characters

α_p	Relaxation parameter for pressure correction equation
β	Volumetric thermal expansion coefficient
Γ	Exchange coefficient (for the diffusion term)
Δ	Liquid fraction correction term in the phase change calculation (Chp. 6)
Δ	Small increment in displacement of time
ΔH	Latent heat energy (Chp. 6) (J/kg)
δ	Small increment in displacement (m) and time (s) (eg. in t, x, y and z directions)
ε	Turbulence dissipation rate of κ (m^2/s)
λ	Arbitrary constant
μ	Dynamic viscosity (Ns/m^2)
κ	Kinematic turbulence kinetic energy (m^2/s)
ν	Laminar kinematic viscosity (m^2/s)
ν_t	Turbulent kinematic viscosity (m^2/s)
ρ	Density (kg/m^3)
ρ_{ref}	Reference density (kg/m^3)
Σ	Summation of a set of values
Φ	Generic dependent variable (Chp. 2 and 3)
ϕ	Conserved scalar variable (volume fraction) for tracking the free surface and also known as the 'metal' fraction in Chp. 6

Superscripts

m,n	Denote value at the previous iteration, sweep or time step
$m+1,n+1$	Denote value at the latest iteration, sweep or time step
new	Denotes value at the latest time step
old	Denotes value at the previous time step
$'$	Denotes a correction term
$*$	Denotes the first intermediate value
$**$	Denotes the second intermediate value

Subscripts

$\boldsymbol{a}$	Denotes the air-phase
e	At East cell-face
h	At High cell-face
l	At Low cell-face
$\boldsymbol{l}$	Denotes the liquid-phase
lim	Denotes a limiting value
$\boldsymbol{m}$	Denotes the mixture of air and liquid

n	At North cell-face
nb	Denotes values of neighbouring cells
s	At South cell-face
w	At West cell-face

Other symbols

D/Dt	Substantial derivative operator, ie. $\partial/\partial t + \underline{u}\nabla$
$\nabla.F$	'div F' = divergence (cartesian)
∇S	'grad S' = gradient (cartesian)
∇^2	Vector operator for second-order derivative
∂	Partial differential
$\mid\ \mid$	Absolute value
$[\![\ ]\!]$	Maximum value of a set of values

Chapter 1
INTRODUCTION AND LITERATURE SURVEY

1.1 The Problem Considered

Free surfaces or boundaries occur in a wide range of physical phenomena which can be observed in nature and many industrial applications. The melting of ice in the sea, the solidification of molten metal in moulds and the seepage of water through dams are just a few examples. All these phenomena share a common feature - the domain of interest is occupied by two fluids. These fluids, such as the most common air-water combination, are separated by a sharp interface or free boundary. Further examples include combinations such as air-petrol in fuel tanks, oil-water and oil-gas found in oil pollution and oil fields respectively, and air-metal as in mould filling. These phenomena, involving free surfaces are generally classified as two-phase flow problems which are not trivial to model mathematically. Phenomena which have more than one free boundary also exist. For example, a combination of air-liquid-solid interfaces occur when filling thin section castings, where solidification can occur simultaneously.

In this study, the author investigated the effects of mould filling on the quality of castings in the foundry industry. A typical control-lever sand casting of an aluminium alloy can be found in *Figures 1.1a and 1.1b.* Common defects such as air porosities and the entrainment of impurities occur when molten metal is filled into the mould cavities with excessive velocity. The metal 'skin' or the air-metal interface consists of an oxide film formed when in contact with air. This film breaks up under high pressure causing the film to be entrained into the bulk of the molten metal. An example can be found in *Figure 1.2*, where a 'blow hole' in a microscope body die was caused by air entrapment. Filling occurs prior to solidification in any casting process (see *Section 1.1.1*). If air and/or impurities are trapped within the fluid during this process then defects are formed during the solidification stage. These defects cost millions of pounds of lost revenue each year to the foundry industry in the United Kingdom alone.

A logical approach to limit this vast waste is to develop reliable predictive tools to assist foundry engineers to control such aspects as the filling velocity. In that way metal can fill the mould cavity smoothly without causing violent splashing. With the advances in

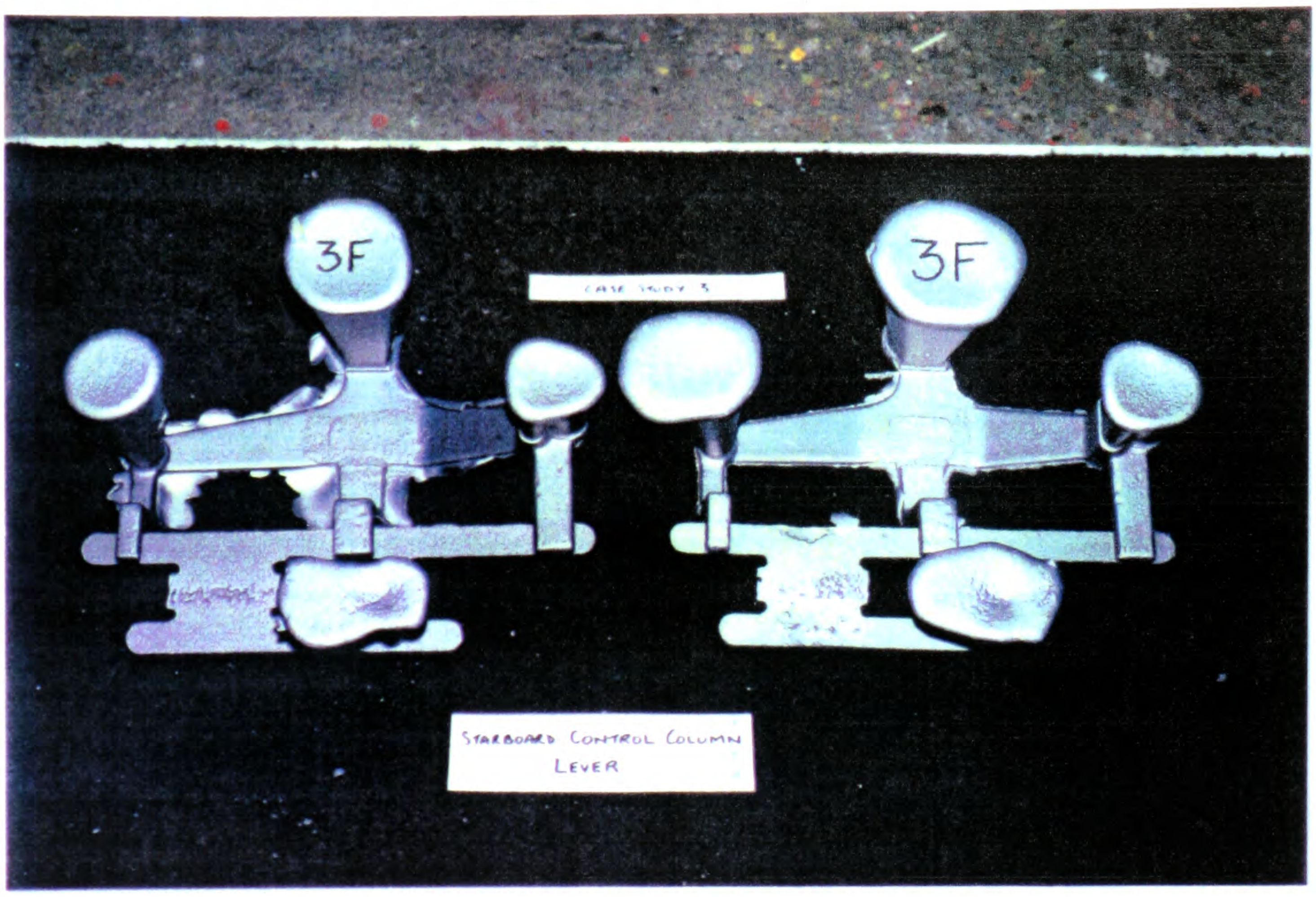


Figure 1.1a Top view of an aluminium alloy casting of a control column lever (*Courtesy of K Preddy*)

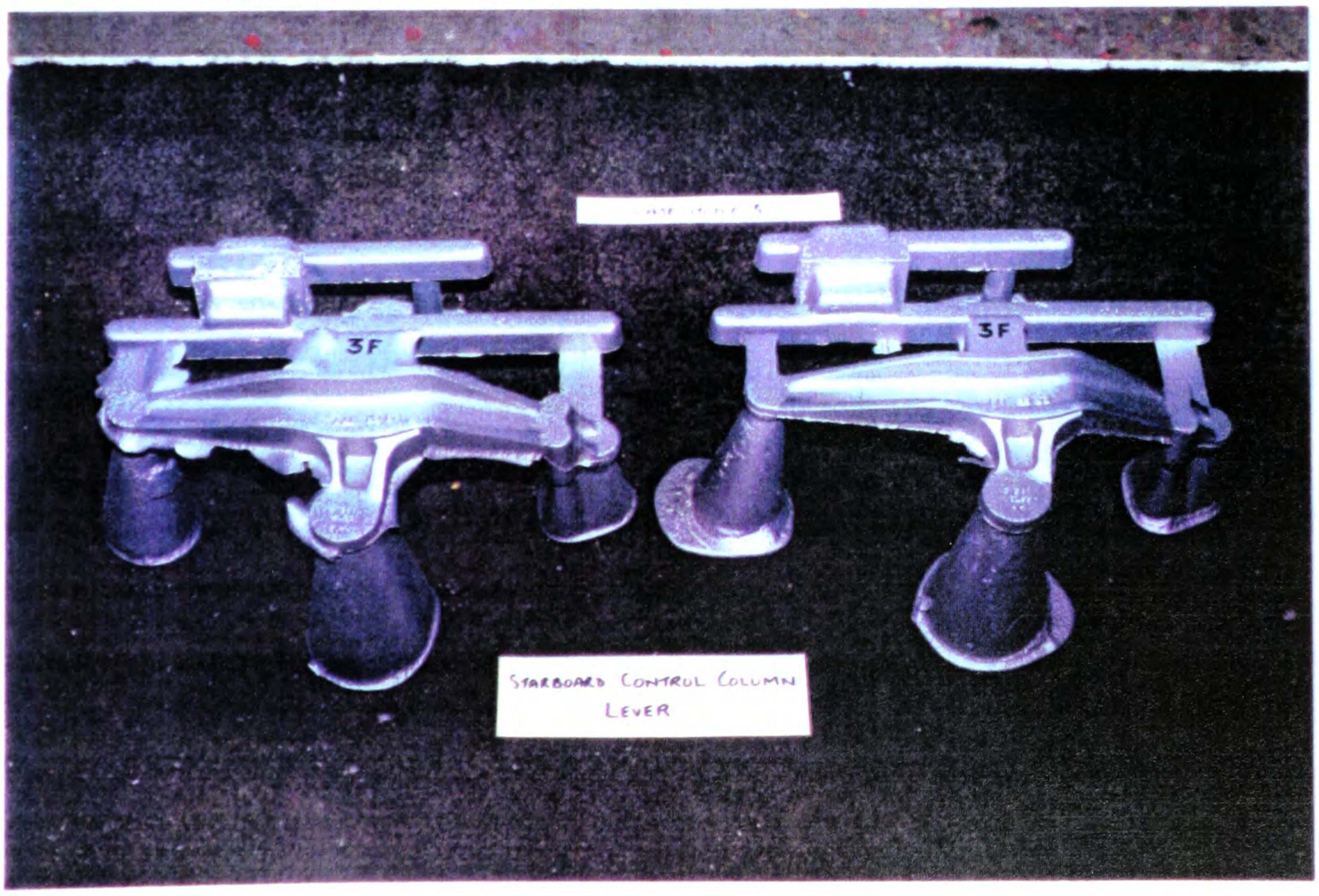


Figure 1.1b Bottom view of the same casting as in Figure 1.1a (*Courtesy of K Preddy*)

computer technology and mathematical modelling techniques in recent years, it is now possible to simulate for example, the *filling* of a casting in a sand mould within hours.

1.1.1 The Quality of Castings and Influence of Modelling Software in the UK

Casting processes have been in existence long before the Bronze Age [Goodway (1988)].

The fundamental procedure for producing castings can be summarised as follows:

(i) melt the metal,

(ii) pour the liquid metal into a prefabricated mould (typically made of sand),

(iii) allow the filled mould to cool until all liquid metal solidifies, and

(iv) break open the mould to recover the solidified casting and carry out secondary treatments, such as fettling and polishing,

(v) The castings are also subject to tertiary treatment such as, heat treatment to release residual stress which has been formed during solidification. See [Campbell (1991)] for more details.

During the entire casting process, a crucial stage of production is *filling*. The quality of casting is primarily determined at the filling and subsequent solidification stages as mentioned before. Most internal defects of casting, such as shrinkages and voids caused by air entrapment are affected directly by the way the mould was filled. In order to produce reliable quality castings, especially for high precision category "one" aerospace components, the running system (see *Figures 1.1a, 1.1b and 1.3*) which carries the liquid metal into the mould must ensure the incoming liquid metal flow is steady and smooth without being turbulent. The running system of a casting generally consists of an *ingate* or *sprue* and one or more *runners* and *feeders* (also known as the risers) as shown in *Figure 1.3*. The shapes, sizes and locations of the sprue and runners control the velocity of the metal flowing into the mould cavity. At the same time, the sizes and locations of the feeders govern whether sufficient additional molten metal are stored to *feed* any shrinkages. The air which exists inside the mould cavity must be expelled completely either via the outlets at the top of the feeders, or through the sand mould itself which is porous.

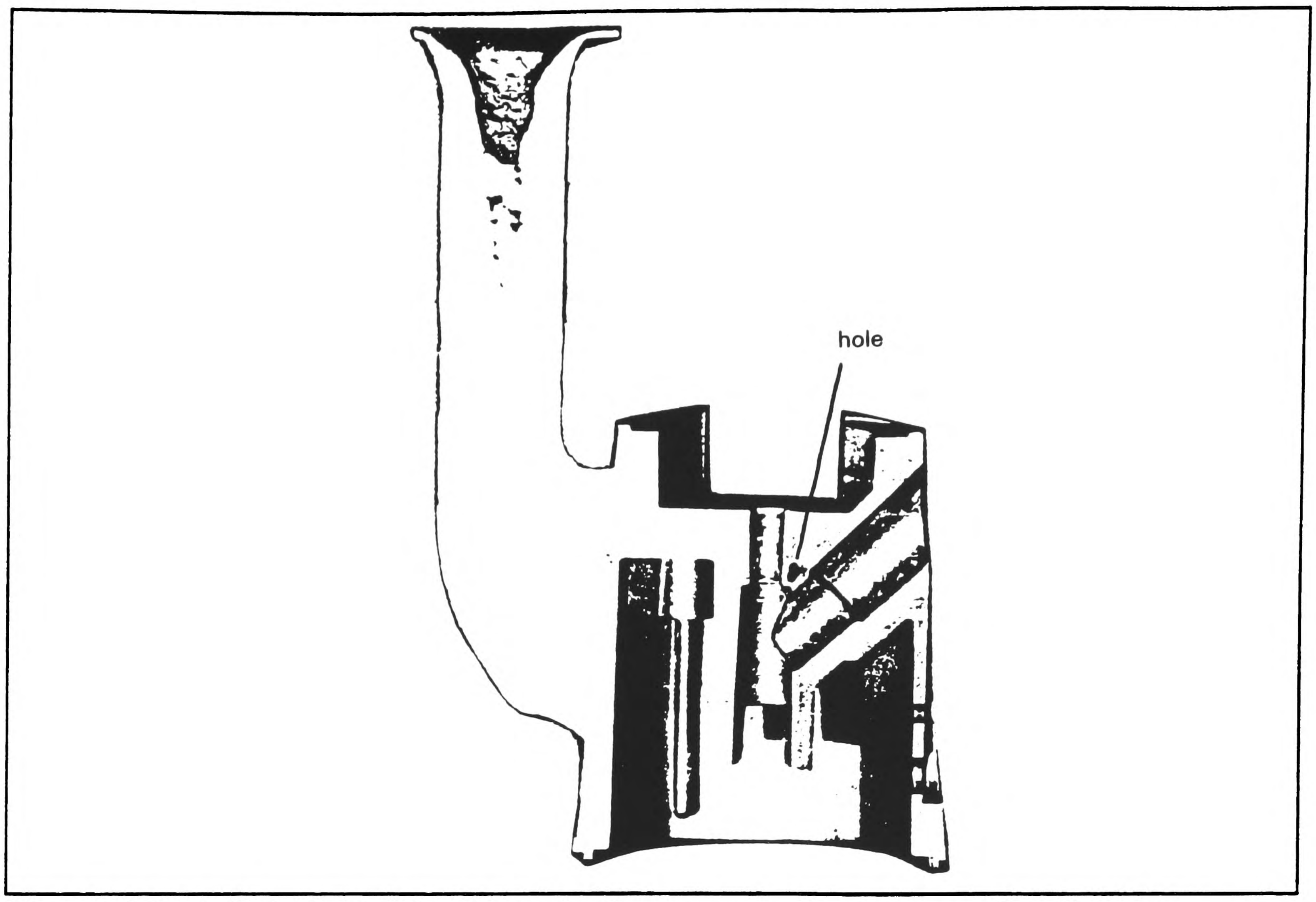


Figure 1.2 A hole is clearly shown in the cross-section of a gravity die cast microscope body.

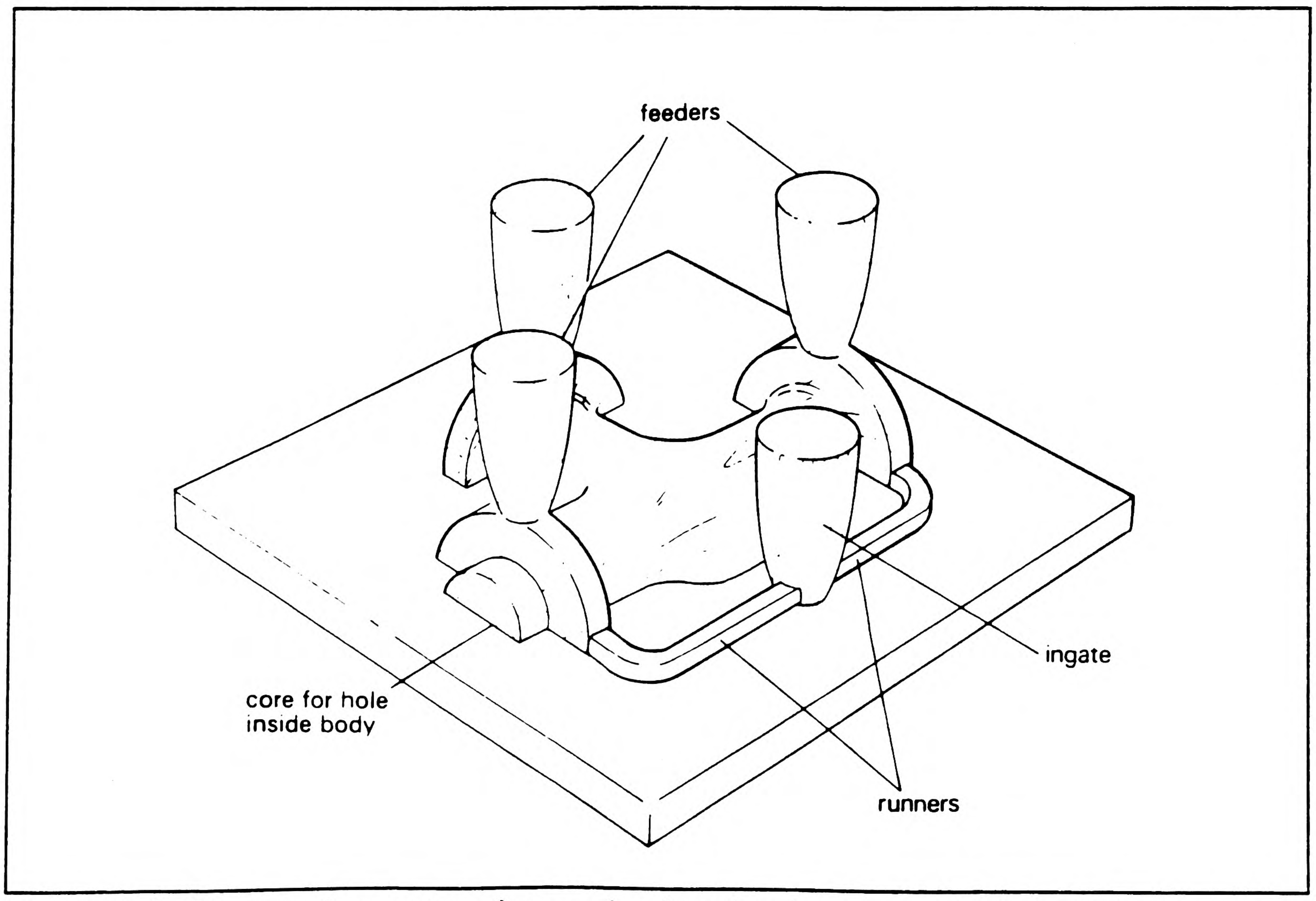


Figure 1.3 The running system for a valve housing is shown which includes an ingate, two runners and three feeders. (Only the top half of the wooden pattern is shown.)

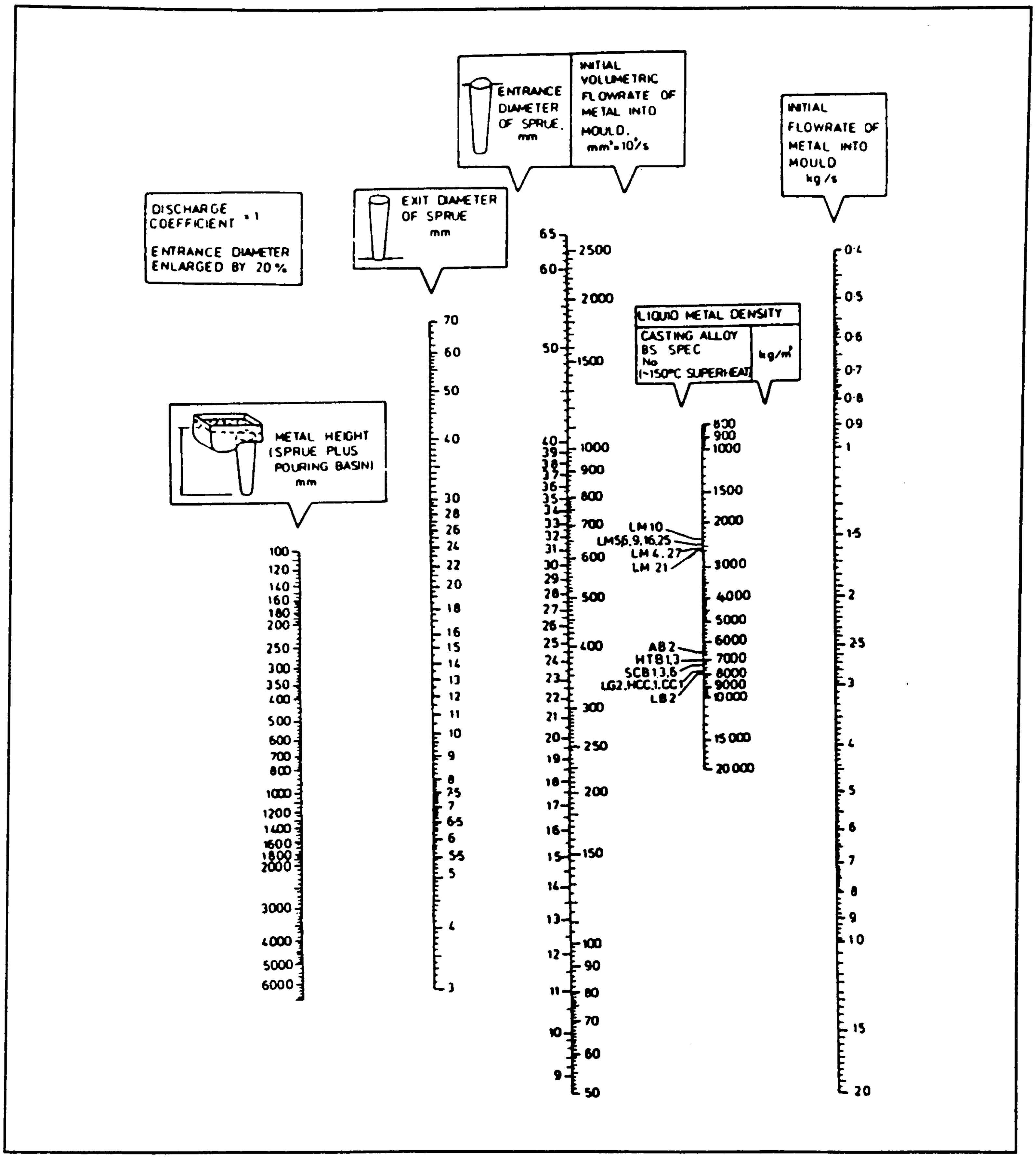
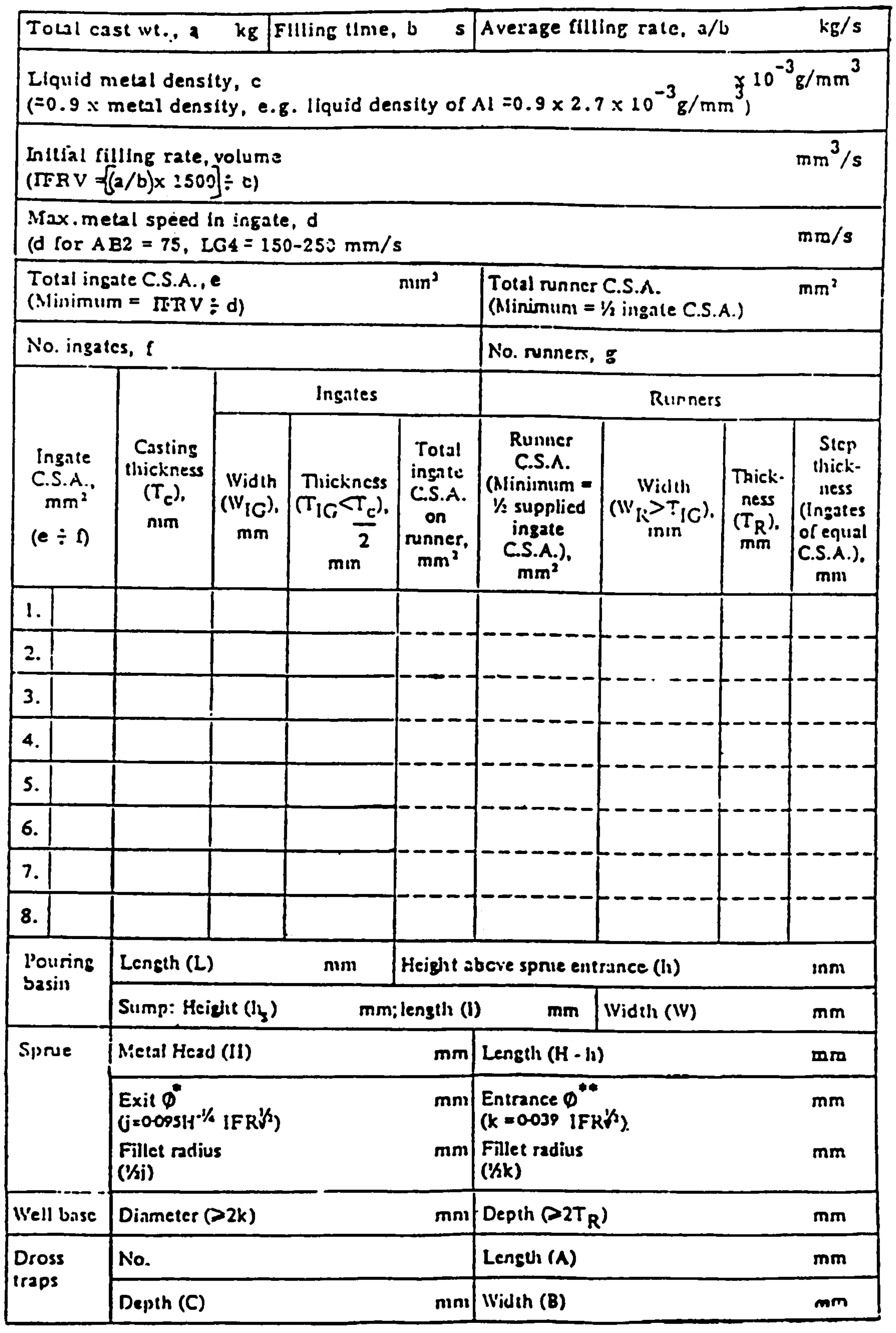


Figure 1.4 A nomogram for the calculation of running system for aluminium alloys.



Total cast wt., a kg	Filling time, b s	Average filling rate, a/b kg/s
Liquid metal density, c (=0.9 x metal density, e.g. liquid density of Al =0.9 x 2.7 x 10^{-3} g/mm^3)		x 10^{-3} g/mm^3
Initial filling rate, volume (IFRV =[(a/b)x 1500] ÷ c)		mm^3/s
Max. metal speed in ingate, d (d for AB2 = 75, LG4 = 150-250 mm/s		mm/s
Total ingate C.S.A., e (Minimum = IFRV ÷ d) mm^2		Total runner C.S.A. (Minimum = ½ ingate C.S.A.) mm^2
No. ingates, f		No. runners, g

		Ingates			Runners			
Ingate C.S.A., mm^2 (e ÷ f)	Casting thickness (T_c), mm	Width (W_{IG}), mm	Thickness (T_{IG}<T_c)/2 mm	Total ingate C.S.A. on runner, mm^2	Runner C.S.A. (Minimum = ½ supplied ingate C.S.A.), mm^2	Width (W_R>T_{IG}), mm	Thick-ness (T_R), mm	Step thick-ness (Ingates of equal C.S.A.), mm
1.								
2.								
3.								
4.								
5.								
6.								
7.								
8.								

Pouring basin	Length (L) mm	Height above sprue entrance (h) mm
	Sump: Height (h_s) mm; length (l) mm	Width (W) mm
Sprue	Metal Head (H) mm	Length (H - h) mm
	Exit Ø* ($j=0.095H^{-1/4}$ IFRV$^{1/2}$) mm	Entrance Ø** ($k=0.039$ IFRV$^{1/2}$) mm
	Fillet radius (½j) mm	Fillet radius (½k) mm
Well base	Diameter (>2k) mm	Depth (>2T_R) mm
Dross traps	No.	Length (A) mm
	Depth (C) mm	Width (B) mm

* Discharge coefficient = 1 ** Enlarged by x 1.2 above minimum

Figure 1.5 A 'work sheet' for the calculation of running system for light alloys.

Wave formation and breaking of the liquid metal surface must be minimised to prevent entrainment of impurities. However, the design of such running systems in the foundry industry is mostly done empirically by trial and error. This kind of production technique is uneconomical in terms of raw material, labour costs, energy and time.

Very few mathematical tools such as nomograms and 'work sheets' (see *Figures 1.4* and *1.5* for examples) are available to aid foundry engineers to calculate the sizes and locations of runners and feeders. Since all these tools are based on empirical rules or formulae, they may only be applicable to certain types of alloy. On the other hand, there are not many computer based software tools available to aid foundry engineers. Most small to medium foundries in UK may have a copy of FeederCalc developed by FOSECO Limited. This software may be used to calculate the running and feeder arrangements for ferrous and aluminium alloys. For bigger foundries, the empirically rule based lattice concept or the so-called 'finite difference' style software such as SOLSTAR [FOSECO and Mircomet] , MAVIS and DIANA [Alphacast Software] are available to carry out 'simulations' for feeding and solidification. However, they do not actually simulate the filling process, since no physical properties and boundary conditions are required. These codes, which they do not work for all alloys, are generally expensive in the range between £8K to £40K.

A few major industrial organisations in the UK, such as Rolls Royce, British Aerospace and Westland Helicopters, can afford the state of art computational-fluid-dynamics (CFD) based casting simulation packages. These software can be used to *simulate* the fundamental physics characterising flow, heat transfer and solidification in moulds. For example, ProCAST [UES Inc], and AF SOLID (2-D) and SYNERGY solidification (3-D) [Dewtec Systems] from the USA, MAGMAsoft [MAGMA] from Germany, SIMULOR [Aluminium Pechiney] from France and CASTEM [Kobe Steel] from Japan are such software packages. However these are expensive, ranging from £20K to more than £50K. However, even these state-of-art software tools may not work well under all situations. The reasons are that further research into fluid and mechanical properties of different kind of alloys and the physics behind the formation of metallic grain structures are still required. Generally software of this type are complicated and need trained experts to operate them correctly and effectively.

1.1.2 The Need of Computer Technology Aid to the UK Foundry Industry

In the long run, mathematical modelling offers a cost effective means of simulating casting processes. Hence, it is vital to the UK foundry industry if it is going to compete successfully in the world-wide market. The dissemination of computer technological aids to the foundry industry is also an important issue, as the industry makes a significant contribution towards the defence and aerospace industries.

At the time of writing, there are a few British based, commercial CFD casting software packages available to the UK foundry industry. One such code is MATRIX (2-D), which is restricted to the simulations of solidification in two-dimensions.

As far as the author understands, there are at least two university based research groups engaging in active research into casting processes in Britain. One group is headed by Professor R W Lewis of University College of Swansea, Wales. They use finite-element formulation to carry out their research. The software package MATRIX is one of their products. Research into filling, solidification, stress, deformation and other related phenomena have also been in progress for some years by a group of researchers at the University of Greenwich, London under the leadership of Professor M Cross. Their research is carried out in the control-volume framework. This thesis forms part of this research.

Besides the research efforts by the two teams above, CFD vendor companies in the UK, such as CHAM Limited and AEA Technology have also incorporated free surface flow capabilities in their general purpose software products. The PHOENICS software code (version 1.6 and later) by CHAM Limited has two options to simulate free surface flows. One method is based on the concept of *height of liquid* [Liu Jun and Spalding (1992)] which is similar to the height function to be described in *Section 1.4.2.1*. The advantage of this method is that it is *implicit* in time (while most methods described in this thesis are *explicit* in time). Hence, its time step size is not restricted by the Courant criterion. However, due to the way that the height of gas and liquid interface is calculated no overturned free surfaces such as convoluted waves, can be simulated. The other method is based on the *scalar equation* concept [Liu Jun (1986a, 1986b), Liu Jun and Spalding (1988)] (described in *Chapter 3*) which has also been adopted by the author to carry out his research in mould filling simulations.

The free surface flow routine developed by R Lonsdale of AEA Technology is similar to the scalar equation method but it uses a heuristic algorithm to determine the sharpness of the gas and liquid interface. This option has been incorporated in AEA Technology's unstructured mesh, non-staggered, control-volume based CFD code known as ASTEC [AEA Technology].

1.2 Practical Relevance

From the mathematical modelling point of view, the physics involved in the casting processes can basically be defined into fluid flow (as in mould filling), heat transfer, phase-change (as in solidification), stress and deformation analyses. All these phenomena are inter-coupled in reality, so the mould filling has a profound influence on the quality of the castings as has been recently reviewed by Campbell (1991).

In addition to mould filling as described above, free surface flow is of particular interest in the aerospace, chemical, petroleum, manufacturing and civil engineering industries. For example, sloshing is encountered in the transportation of liquid fuels in aircraft (eg. external fuel tanks) and road tankers. If excessive sloshing occurs in these vehicles, extreme dynamic loads may be imposed onto the structure of the fuel tank. The design engineers must therefore, take free surface motion within fuel tanks into account when designing the structural requirements for these vehicles. Further examples of the applications of free surface flow modelling include:

- The injection moulding of plastics (or polymers), instead of metals.
- The impact of tidal waves on the structure of dams or other coastal defence due to high sea level or even earthquakes.
- The filling and emptying of liquid in a vessel under gravity.

- The rapid expansion of a vapour bubble trapped in a nuclear reactor fuel compartment, or in the cooling system of a pressurised water cool nuclear reactor (PWR).
- Simulation of froth flotation process, for separating minerals from unwanted species in mined ores, in a cyclonic flotation machine.
- Simulation of gas slugs through an oil pipeline.
- Simulation of seepage of ground water through dams which involves modelling unsaturated flow through porous media.
- Separation of lighter oil from heavier ones in an oil refining process.

1.3 Objectives

The present study aims to achieve the following objectives:

1. To develop and validate a free surface flow model suitable for simulating the mould filling processes in three-dimensions (3-D).
2. To investigate different methods to improve the computational efficiency of the 3-D filling model.
3. To investigate the effect of filling on heat transfer and solidification of liquid metals and vice versa.

These are achieved by means of:

1. conducting a comprehensive literature survey on free surface flow methods to establish the best approach.
2. collaborating with BNF-Fulmer Metals Technology to obtain experimental data to validate the proposed filling model.
3. comparing simulated results with those published by other researchers in the literature to establish relative accuracy and efficiency.
4. implementing the filling model into a parallelised CFD code as an attached module, so that it can be run in parallel in a network of processors to enable large low-cost simulations.

1.4 Review of Literature

A distinctive feature of free surface flows is the presence of a free boundary which separates one fluid from another, for example oil floating on top of water. From a mathematical perspective, most free surface flow phenomena can be classified as moving boundary (Stefan) problems, since their boundaries vary with time, according to Crank (1984). So the problem of mould-filling belongs to this classification.

Two major difficulties arise in modelling free surface flows. Firstly, the evolution of free surface boundary in time needs to be tracked or captured, and secondly the determination of the flow field in the domain. Since, in the case of mould filling with air taken into consideration, the density ratio between air and liquid metals is large (a factor of a thousand), this results in an enormous mass discontinuity at the air-liquid interface. Hence, the use of mass balance to track the free surface boundary is not possible in a single-phase framework. However, if air is neglected in the model, single phase methods such as Marker And Cell (MAC) and fractional Volume Of Fluid (VOF), may be used to simulate free surface flow. But it would also mean that certain physical phenomena, such as air entrainment cannot be simulated without using additional approximations for the air pressure.

Before proceeding to the review, it is useful to be aware that the computational frameworks to which these techniques can attach to may be classified in three major types. These are finite-difference (FD) and its close relation: finite-volume (or control-volume (CV)) methods, finite-element (FE) methods, and boundary-element (BE) or boundary-integral methods.

The review presented here focuses on methods used by CFD practitioners on mould filling and related applications. The work reported here mainly concentrates on the control volume framework, less emphasis will be placed on either FE or BE based methods. However, readers who are interested in the general mathematical formulations of numerical methods for free surface flows in FD, FE and BE frameworks should see an extensive review given by Yeung (1982). And for those who are interested in tracking methods for modelling air bubble motions may find a review by Unverdi and Tryggvason (1992) useful.

1.4.1 Eulerian or Lagrangian Approach?

Since the 1960's, more and more research effort has been invested on finding a better numerical method to tackle the difficulties mentioned above.

In general two main techniques are used by researchers; one retains a fixed grid for the computational domain, whilst the other modifies the grid in a Lagrangian manner to match the distortion of the interface. Zhang et al (1993) used the Lagrangian approach to carry out simulation on filling and solidification. The advantage of the Lagrangian approach is that a sharp interface, free from numerical smearing, can be obtained. However, the major disadvantage is the requirement of continuously re-meshing the grid to match the interface. Otherwise highly distorted grids producing non-smooth and deformed surfaces will be resulted. The Lagrangian approach is also expensive in CPU time.

For the fixed grid (or Eulerian) approach , the free surface interface is tracked by methods which are to be described in later sections. The majority of researchers favour this approach, with the prominent being, Harlow and Welch (1965), Hirt and Nichols (1981), Liu Jun (1986a, 1986b), Hwang and Stoehr (1987), and Lewis et al (1991a, 1991b). The main disadvantage of the Eulerian approach is the numerical smearing at the interface as mentioned earlier.

In the following subsections, three major categories of computational techniques available for tracking free surface boundaries will be reviewed. They are the front tracking and capturing methods, volume tracking methods, and moving grid methods. The first two categories are Eulerian based methods and the latter is Lagrangian based. Of course, other new methods will also be reviewed.

1.4.2 Front Tracking and Capturing Methods

The fundamental idea of *front tracking* (also known as surface tracking) methods is quite straightforward. Imagine that a sufficient number of coloured small polystyrene balls have been scattered in a water tank. The wave motion traversing the tank can be outlined by the movement of these balls. In terms of mathematics, the interface between for example, liquid and air, can be specified by an ordered set of particles or imaginary points. The position of the interface can then be approximated by interpolation between these points. A string and balls is sufficient to represent a free surface in two dimensions; and a 'net' of balls can represent a free surface in three dimensions [Maxwell (1977), Liu Jun (1986b)].

Alternatively, in *front capturing* methods, the basic approach involves using a high order 'shock-capturing' scheme (usually of order two), such as van Leer TVD (total variance diminishing) scheme [van Leer (1977)] and Quick [Leonard (1980)], to approximate the convective fluxes that transport across the control volume cell faces. These shock-capturing schemes enforce a monotonicity condition (see [van Leer (1977)] for a definition) explicitly at the free surface front, and suppress the numerical smearing. According to a review by Unverdi and Tryggvason (1992), shock-capturing methods generally serve well for shock wave discontinuities encountered in aerodynamics, but less well for contact discontinuities, such as crack propagation. Also, a relatively fine grid is needed to give good resolution, therefore its application is usually restricted to unsteady flow simulation in two-dimensions.

The advantage offered by both front tracking and front capturing, over other Eulerian based methods is that higher accuracy or sharper fronts can be obtained. But for the front tracking methods, considerably higher computational cost and complexity in implementation are the penalties to be paid, since additional computational elements are required to track the front. Techniques for front tracking schemes are described below.

1.4.2.1 Height Function Methods

Front tracking in its simplest form can be achieved by using a height function, according to reviews by Hirt and Nichols (1981) and Liu Jun (1986b) respectively. The relative position of a fictitious point may be approximated by its distance from a reference surface through a height function. For example, the height *H* of the free surface in a rectangular mesh of cell width δx and height δy, above the bottom of the mesh in each column of cells can be defined by a function of the form $H = f(x,t)$. A curve can be approximated by assigning values of *H* to discrete points of x. For free surface flows, the height function is governed by a kinetic equation (*Equation 1.1*) which ensures the interface moves with the local flow field only.

$$\frac{\partial H}{\partial t} + u\frac{\partial H}{\partial x} = v \tag{1.1}$$

where *u* and *v* are velocities obtained after solving the Navier-Stokes equations (see *Chapter* 2) along the x and y directions respectively. Hirt et al (1975) adopted this method to determine the shape and location of the interface, as part of the development of the well known Solution Algorithm - SOLA technique. Miyata and Nishimura (1985) used this method to calculate sea waves in a study for the design of ship hulls.

However, the major drawbacks of this technique is that it does not work well when the boundary gradient, *dH/dx*, exceeds the aspect ratio $\delta y/\delta x$, and cannot handle multi-valued surface having more than one y value for a given *x*, such as convoluted waves [Hirt and Nichols (1981)]. On the other hand, the height function method only requires a single array to store the height values and so is much more economical, with storage overheads than the particle tracking methods. Since only the height values need to be updated in each solution cycle, programming is simpler than for the particle tracking methods. The height function method can be readily extended to three-dimensions [Nichols and Hirt (1973)].

1.4.2.2 Particle Tracking Methods

Similar to the polystyrene balls analogy above (*Section 1.4.2*), a string of massless, fictitious particles are scattered evenly to mark the free surface between the fluids at the beginning of the calculation. Each particle is identified by its location in coordinates x_p and y_p, with its local velocity components represented by u_p and v_p respectively. The movement of the particles is governed by a set of Lagrangian equations as follows:

$$\frac{\partial x_p}{\partial t} = u_p \tag{1.2}$$

$$\frac{\partial y_p}{\partial t} = v_p \tag{1.3}$$

Using an explicit integration on *Equations 1.2* and *1.3* , a set of explicit algebraic equations can be obtained as follows:

$$x_p^{m+1} = x_p^m + u_p \delta t \tag{1.4}$$

$$y_p^{m+1} = y_p^m + v_p \delta t \tag{1.5}$$

where δt is the time step size. The superscripts *m* and *m+1* denote the values at the beginning and end of a time step respectively.

At the beginning of each time step, the density distribution of the fluid is determined from the position of the free surface. This density field is then used in the solution of the Navier-Stokes equations for velocities. The local velocities (u_p and v_p) of a particle can be interpolated from the four neighbouring velocities at the cell faces. At the end of the time step, the positions of these particles are updated according to *Equations 1.4* and *1.5*, in a Lagrangian fashion. These particles are linearly connected and ordered. Intermediate particles can be added or removed from the surface if the distance between two adjacent particles is less or greater than the respective prescribed limits.

Maxwell (1977), and later Liu Jun (1986b), showed that the method works well for the rise of bubbles through a liquid, motion of water waves, their breaking over a surface, and the collapse of a water column in two-dimensions. With the aid of GALA [Spalding (1974, 1977)] to ease numerical difficulty due to a large change in density across the interface, Maxwell and Liu Jun solved the equations of momentum in both liquid and gas phases. More recently, the same method was employed by Villasenor (1988) and similarly by Castrejon (1984) who adopted it as a flow visualisation tool to his work on natural convection in an annular cylinder.

As far as the implementation is concerned, the level of programming skill (in FORTRAN) and computer storage required to manage the link-list of particle locations will be quite demanding in two-dimensions, and daunting in three-dimensions. Furthermore, the necessity of housing-keeping at each time step, together with the restriction of Courant criterion on the size of the time step will inevitably lead to large CPU time requirements.

Overall, particle tracking methods give a finer resolution of details at free surface than volume tracking methods [Liu Jun (1986b)]. However, it suffers from a serious weakness. When two surfaces intersect, or a surface folds over itself as in a convoluted wave, the ordering of the particles must be re-organised. This re-ordering process is very complicated, if not impossible in two-dimensions [Liu Jun (1986b)]. So for practical mould filling simulations in three-dimensions, the particle tracking method is not viable.

1.4.3 Volume Tracking Methods

In this category of methods, the interface is specified by a common boundary of two adjacent fluid regions. The region and its actual surface shape is dependent on the distribution of fluid markers.

1.4.3.1 The Marker and Cell Method

The Marker and Cell (MAC) Method is one of the earliest and well-known volume tracking methods. It was first proposed by Harlow and Welch (1965) as a numerical technique to solve for time-dependent, viscous and incompressible fluid flow with free surfaces; and subsequently appeared in a Los Alamos technical report [Welch et al (1966)].

Description of the technique

This method is essentially a hybrid of the Eulerian and Lagrangian approaches as reported by Hwang and Stoehr (1987). The finite difference equations of the MAC method is written in an explicit form. The computational domain is divided into a number of cells which remain at rest, as in the Eulerian approach. In addition, a number of massless particles or Lagrangian markers are initially scattered to identify the fluid region. These markers are then moved through the Eulerian mesh in a Lagrangian manner, depending on the local flow field in the Eulerian cells. According to the distribution of these fluid markers in the Eulerian mesh which can be differentiated between *interior*, *surface* and *exterior* regions. The presence or absence of these markers in a cell indicates whether a cell is *full* or *empty*. If a full cell is adjacent to an empty one then this cell is known as a *surface* cell. Free surface boundaries between two fluids, such as air and liquid metal, can then be defined by the line segments joining markers in these surface cells in a sequential order [Nichols and Hirt (1971)].

Due to the Eulerian-Lagrangian nature of the MAC method, it treats the interior and surface regions separately when solving for the momentum equations. In the interior or fluid region, the calculation for the new flow field at the *n+1* time step must both satisfy the conservation of momentum and mass, by ensuring the velocity divergence reduces to zero, simultaneously. The velocity divergence is related to the pressure field which is calculated by solving a Poisson equation. In the surface region, the new velocity components are calculated based on the conservation of momentum on the filled sides of the surface cells. And the velocity field is extrapolated to the empty sides of the surface cells to approximate the stress conditions at the free surface. After the new velocity field is computed, the markers are transported accordingly and form a new fluid configuration. The process is repeated until the desired number of time steps is reached [Hwang and Stoehr (1987)].

The readers should note that when using the MAC method to calculate transient flows with free surface, it requires extra calculations in two areas. Firstly the determination of marker locations, and secondly the determination of free surface boundary conditions. The boundary conditions are calculated by ensuring that the normal stress equals the external pressure, and the tangential stress at the boundary equals to zero. Also, the markers are not used in the solution of the Navier-Stokes equations [Harlow and Welch (1965)]. In other words, the fluid density is not determined via the distribution of these markers.

Applications of the MAC method

The first application of the MAC method was the collapse of water column [Harlow and Welch (1965)]. Successive improvements to the crude approximations of the normal and tangential boundary conditions, as appeared in the original method, have been carried out by Hirt and Shannon (1968), Chan and Street (1970), and Nichols and Hirt (1971) respectively. To illustrate the effect of these improvements, Nichols and Hirt (1971) carried out a number of typical free surface flow comparisons with experimental data and those of the original method. Good agreements between experimental data and their results were reported. In addition, a simulation of a splashing drop (of liquid) falling in a deep pool was presented to demonstrate the MAC method's ability to calculate highly contorted and colliding surfaces.

The constraints on the numerical stability of the MAC method when applied to flow problems with low Reynold numbers was investigated by Pracht (1971). Pracht concluded that the explicitly formulated MAC method is restricted to flow problems with Reynold number greater than one. In other words, it means that the explicit MAC method is unstable when dealing with very viscous flows. Pracht devised an implicit formulation for the MAC method to overcome this restriction. Although, the implicit scheme is unconditionally stable, it was later found by Deville (1974) to be time-consuming. This fact encouraged Deville (1974) to experiment with the applicability of four finite-difference schemes within the framework of the MAC method. Assessing from aspects of efficiency, accuracy and ease of programming, he proposed that the ADI Douglas-Rachford scheme should be adopted for the numerical integration of momentum equations for low Reynold number flows.

The MAC method was extended to three-dimensions by Hirt and Cook (1972) to model low-speed fluid flow around block-type structures.

A study of unsteady, free surface flow of a fluid being pumped upward from a tube in the base of a vertical cylindrical cavity was performed by Smith and Wilkes (1975). They used the MAC method in conjunction with a Dufort-Frankel (1953) approximation in polar coordinates to model this scenario. The comparisons between simulated results and those of experimental data were encouraging.

Ramshaw and Trapp (1976) adopted the MAC method with the Chorin-Hirt modification [Hirt and Cook (1972)] and a donor-acceptor differencing scheme to model the sloshing of vapour and liquid in a closed cavity. The interesting points about this work are that, firstly a transient flow of homogeneous two-phase (gas-liquid) fluid was modelled in a one-phase framework. Secondly, the donor-acceptor technique was used to combat numerical smearing at the fluid interface.

Hwang and Stoehr (1983, 1987) adopted the MAC method to simulate mould fillings of water in two-dimensions. The flow patterns calculated by the method in the filling of a squared cavity were compared with water experiments which indicated that the results were consistent [Hwang and Stoehr (1987)].

Variants of MAC

Many variants of the MAC method have been developed, including the Simplified MAC (SMAC) method by Amsden and Harlow (1970), and recently SOLA-MAC in three-dimensions by Lin and Hwang (1989).

• Simplified MAC

The major advantage of the SMAC method over the basic MAC method is that the boundary conditions for the Poisson equation for pressure are strictly homogenous. The basic difference between the MAC and SMAC methods is that the pressure field is not directly solved for the latter. Therefore, the SMAC method has fewer boundary conditions to complicate the programming logic. Hence, the SMAC method can be solved more efficiently, thus making saving in CPU time. A number of free surface flow problems, such as the sloshing of water in a tank, were modelled using this technique. The results were in good agreements with experimental data [Amsden and Harlow (1970)].

Anzai and Uchida applied a modified SMAC method, in quasi-three-dimensions, to investigate the filling patterns of flat plate die castings. Their method has incorporated features found in an empirical gapwise direction mould filling model proposed by Frommer (1933). In Frommer's model the mould cavity is filled from the opposite side of the gate and the final filled position is in the vicinity of the gate. Anzai and Uchida made the following modifications to the SMAC method:

(1) The fluid keeps its thickness as it flows into the cavity from a thinner cavity element, such as the gate.

(2) An additional cell type FS is introduced in the calculation to account for the effect of the free surface perpendicular to the gapwise direction.

• Algorithm for Porous Flows

Casulli and Greenspan (1982) developed an algorithm for modelling general free surface, porous flow problems. They combined the MAC method with the Courant-Isascson-Rees method (1952) which was developed for nonlinear hyperbolic differential equations. The algorithm was tested on a steady-state dam flow problem. They claimed that the new method is more powerful than either of its constituents and was fast, economical and accurate.

• SOLA-MAC

The Solution Algorithm - MAC (SOLA-MAC) method was developed by Lin and Hwang (1989) to model mould filling in three-dimensions. This is a hybrid technique which employs the SOLA-VOF algorithm [Hirt and Nichols (1981)] to solve for the Navier-Stokes equations, but instead of using the volume fraction to track the free surface, it uses marker particles as in the MAC method. The method was tested on a simple block casting and a casting of a preformed forging die. The results were claimed to be satisfactory. However, it requires a rearrangement scheme to act as a memory manager for the marker particles. Similar to other particle based algorithms, the requirement of heavy storage for marker particles may prove to be a potential problem, if large scale, multi-physics simulation is to be carried out.

The SOLA-MAC technique has also been applied to a simulation of filling in a rotating vertical centrifugal casting [Hwang J D et al (1991)].

Comments on MAC method

The MAC method does not suffer from the problem of surface interactions which is associated with particle tracking methods. Hence, the programming logic required to set up a such code is simpler. But on the other hand, the MAC method as with other volume tracking methods are less capable in giving high resolution of details at the free surface. A serious weakness of the MAC method and its variants is that it needs memory to store coordinates of marker particles. Disregarding whether the problem is to be modelled in two- or three-dimensions, this constraint may pose serious limitation on the size of problems, if finer mesh or multi-physics such as, heat transfer and phase change are to be included. Furthermore, additional calculations or house-keeping routines are needed to determine the exact location of markers, the shape of the interface, and to update markers to their new positions. This extra work also takes up expensive CPU time.

1.4.3.2 The Volume of Fluid Method

The Volume of Fluid (VOF) method was developed by Hirt and Nichols (1981) at the Los Alamos Scientific Laboratory in 1979. This is one of the most popular algorithms developed for mould filling simulations. Recently, it has been adopted in a number of commercial casting simulation software codes such as, ProCAST [UES Inc], SIMULOR [Aluminium Pechiney] and MAGMAsoft [MAGMA]. Also, there are no less than ten papers in the proceedings of the conference on Modelling of Casting, Welding and Advanced Solidification Processes VI (1993) which use the VOF method.

The essence of the VOF method is that it relies on a volume or concentration fraction, F, to determine whether a cell is full or empty of liquid. F is a step function which takes the value of one for full of liquid and zero otherwise. The free surface boundary is contained in cells where their values are between zero and one.

The volume fraction F in each cell is updated during each solution cycle according to the advection equation below:

$$\frac{\partial F}{\partial t} + u\frac{\partial F}{\partial x} + v\frac{\partial F}{\partial y} = 0 \qquad (1.6)$$

Equation 1.6 states that F moves according to the fluid velocities. In a Lagrangian grid, this equation simply means that F remains constant at each cell. In such cases, F serves as a stationary marker to indicate the state of the cell, whether it is full or empty.

In an Eulerian grid, the flux of F moving with the fluid through the cell faces must be calculated. Since F is basically a discontinuous, step function (between zero and one), numerical smearing at the free surface will occur if standard differencing scheme (such as, upwind and hybrid) were used. Therefore, a differencing scheme which can preserve F's discontinuous nature must be used. A donor-acceptor approximation [Johnson W E (1970)] was adopted in the original VOF derivation by Hirt and Nichols (1981).

In order to impose accurate boundary conditions to the free surface, the exact location of this interface within a surface cell is required. The interface is reconstructed cell-wise at each time step by some reconstruction schemes [Chorin (1980)] which act on the volume fraction F of the cell under consideration and its neighbouring cells.

The VOF methods and its variants (such as the SOLA-VOF method) have been successfully applied to flow situations in which more than one moving interface exists. Nichols and Hirt (1981) have tested the method on six test cases, for which either experimental data or analytical solutions are available for comparisons with the simulated results. The VOF methods was first embedded in the SOLA-VOF code [Nichols and Hirt (1981)] which later evolved to become the well known commercial filling simulation software code *FLOW-3D* [Flow Science Inc].

Partom (1987) applied the VOF method to model the sloshing of fluid in a partially filled cylindrical container.

Golafshani (1988) developed a technique, which is an extension of SOLA-VOF, to investigate incompressible flow in the low-Reynolds-number range, ie. creep flows. His technique is known as *cell iterative adjustment technique* (CIAT).

Lin and Hwang (1988) applied the SOLA-VOF method to investigate filling of castings with heat transfer analysis in two dimensions.

Mishima and Szekely (1989) developed a model using the SOLA-VOF method with heat transfer and solidification to simulate the filling of cylindrical moulds.

Backer and Sant (1991) used the FLOW-3D code [Flow Science Inc] to investigate the formation of 'pinholes' in iron castings. Pinholes are caused by the entrainment of iron silicate particles or gas bubbles in the iron castings. The kinetic behaviour of these particles and bubbles during mould filling can be simulated by the Basset-Boussinesq-Oseen (BBO) equation provided that the flow field is known in advance.

Tomiyama and Sou (1991) applied the VOF method to simulate the rising of a single bubble in liquids.

Lemos (1992) developed a code known as 2D-HYDROTUR to model turbulent flows with free surfaces. His code is an extension to the SOLA-VOF method.

1.4.4 Moving Grid Methods

In moving grid methods, the original grid system is subject to constant adjustment to approximate the free surface interface. The governing equations are then solved on the new distorted mesh ,and the grid points on the interface are treated as a moving boundary. In this way, the undesirable numerical mixing or smearing between the different fluid regions is reduced or eradicated.

Local Adjustment Methods

Disregarding whether a front or volume tracking method is used, if the reconstructed free surface is continuous, then an adjustment in the underlying grid location can be made in the vicinity of the interface at each time step. The idea is to move those grid points which are within a half-cell spacing of the interface to the nearest position horizontally or vertically where the interface intersects a grid line. Thus, the cell edge and diagonals on the new moderately distorted grid is approximating the interface. Hyman and Larrouturou (1982) adopted such a method for their study. Alternatively, the concept of cell 'porosity' or blockage factor can be used to adjust the grid boundary to match the free surface interface [Pericleous and Rhodes (1984), Liu Jun (1985), and Simitovic et al (1986)].

Lagrangian Methods

The initial region of fluid is subdivided into a number of discrete elements with cell boundaries aligned with the interfaces. The Navier-Stokes equations are then solved in a cell by cell basis, whereby the cell edges are all treated as part of the interface. Hence, they can track the interfaces. As new interfaces are created, new cells are added or existing cells re-adjusted. On the other hand, if the volume of existing cells tend to zero, they are merged or deleted. In this way, the numerical diffusion across the interface is eliminated, and the varying profile of the interface that moves with the fluid can be tracked. Zhang et al (1993) used an arbitrary Lagrangian-Eulerian (ALE) Petrov-Galerkin finite element technique to study mould filling phenomena. Their method uses triangular elements to form the interface in the way as described above.

1.4.5 Other Tracking Methods

1.4.5.1 The Scalar Equation Algorithm

The Scalar Equation Algorithm (SEA) is similar in approach to the VOF technique. In both methods a scalar advection equation, which determines the volume fraction of each cell in the domain of interest, is employed to track the position of the interface. However, the SEA method covers both the air and liquid phases when solving for the momentum equation unlike the VOF method which only covers the liquid phase. The SEA method is used by the author in his study of mould filling phenomena.

The SEA method was developed originally by Liu Jun at Imperial College [Liu Jun (1986a,1986b), Liu Jun and Spalding (1988)] and applied to two dimensional problems involving more than one fluid layer. The details of formulation of the method in three dimensions can be found in *Chapter 3.*

The author and co-workers have applied this method to simulate mould filling problems in three-dimensions [Chan et al (1991), Pericleous et al (1993)]. The results of some simulations can be found in *Chapter 4.* Jeurissen and Eldijk (1991) in collaboration with CHAM Limited have applied the SEA technique to simulate the filling of liquid into a circular vessel against gravity. The results were validated against experimental data. At the time of writing, the SEA technique has been incorporated by Liu Jun (of CHAM Limited) as one of the free surface modelling options within the latest version of PHOENICS CFD code [CHAM Limited]. A summary of weaknesses of this technique can be found in *Chapter 7.*

1.4.5.2 Variable Blockage Methods

When two fluids have a large variation in density, such as air and water, the influence of air on the flow field of the liquid can often be neglected under many practical situations. Therefore, the air phase in the problem domain is not calculated. Instead a 'uniform' or constant pressure boundary at the free surface is assumed. The fixed computational domain, which extends to both above and below the interface, is being used to solve for the liquid phase alone. The free surface representing the interface between the two fluids, can be defined in the form of *blocked cells* or *blockages* using a volume porosity factor at each cell. The blockage factor (r) has limits from zero to one, which indicate whether the cell is fully occupied by liquid ($r=0$) or air ($r=1$). The cells with blockage factors between zero and one must therefore contain the free surface interface.

The solution of the problem can be obtained in an iterative manner as follows:

(1) The interface shape is guessed, for example as being planar.

(2) The Navier-Stokes equations are solved, with the result that a pressure distribution is computed for the cells through which the interface passes.

(3) The interface cells for which the pressure exceeds the atmospheric pressure by an amount δp, reduces the blockage factors, which in turn

raises the interface position. For interface cells in which δp is negative, the blockages are increased. This implies that the interface height is reduced. The height changes are made equal to $\delta p/(\rho g)$, in accordance with hydrostatic laws.

(4) The Navier-Stokes equations are solved again, yielding a new set of interface-cell pressures, which are usually found to be closer to the prescribed atmospheric value.

(5) The process of interface adjustment is repeated until the solution is converged.

Pericleous and Rhodes (1984) have successfully applied this technique to investigate the performance of a cyclonic flotation machine which is used in mining industries to separate valuable minerals from unwanted species in mined ores.

Liu Jun (1985) adopted a similar technique to predict the free surface of a steady flow over curved beds.

Simitovic et al (1986) applied the same technique to simulate the giant waves following the collapse of a dam.

This technique is easy to understand and can be implemented in two- and three-dimensions. Furthermore, it is implicit in time and hence no restriction on the time step size for numerical stability is required. However, this technique becomes invalid in situations where the surface height becomes multi-valued, for example a water wave that falls back onto itself.

1.4.5.3 Two Fluid Technique

This technique has been designed to study the flow of two phases, one of which is usually continuous and the other dispersed; for example particles in liquid, droplets in air, and air bubbles in fluid. This method relies on the solution of a coupled set of Navier-Stokes equations one for each phase, together with a volume conservation equation. Two phase algorithms such as the IPSA [Spalding (1977,1980a)] algorithm can be used together with GALA [Spalding (1974, 1977)] and a donor-acceptor type flux approximation [Ramshaw and Trapp (1976)] to ensure the interface remains sharp. Mould filling and sloshing problems can be solved by this technique. Aldham et al (1982) applied this technique to model the effects of gas bubble evolution following an explosion in a nuclear reactor containment vessel.

The advantage of two-phase techniques is that not can only the free surface movement be tracked, but also wave breaking and atomisation phenomena can be accommodated. However, the method is complex and requires inter-phase momentum, heat and mass transfer correlations to be specified. It is also expensive in terms of CPU time and storage, due to the large number of equations involved.

1.4.5.4 Vortex Methods

Chen and Vorus (1992) applied a vortex method to simulate free surface flows in marine applications. Vortex methods have been used extensively in modelling aerodynamic lifting bodies. Their method is a potential flow formulation which uses the exact non-linear free surface boundary condition at the exact location of the instantaneous free surface. The position of the free surface, on which vortices are distributed, is updated using a Lagrangian scheme following the fluid particles on the free surface. They reported that the method is particularly suitable for studying non-linear unsteady interactions. For examples, the body-wave interaction and body-wake-wave interaction problems.

1.4.5.5 Enthalpy based VOF method

Recently, Swaminathan and Voller (1993) proposed a new variant of the VOF method based on an 'enthalpy' type variable. This method is free of numerical smearing and is not restricted by the Courant criterion. However, a serious drawback of this method is that once a cell is filled with fluid, it cannot be emptied later. Therefore, it is not possible to simulate convoluted waves or sloshing of fluid at the moment.

1.4.5.6 The Height of Liquid Method

Recently, Liu Jun and Spalding (1992) developed a free surface model based on the ideas of height functions and the GALA algorithm [Spalding (1974, 1977)]. The Height of Liquid (HOL) method is implicit in time and it does not seem to suffer numerical smearing at the interface. However, it cannot be used to simulate situations, such as the sloshing of fluid, where convoluted waves exist. This is because the HOL method inherited the same drawback suffered by the height function methods (*Section 1.4.2.1*). No formal publications of this method can be found in the literature yet. Nevertheless, it is included in the latest version (1.6.6) of PHOENICS CFD code [CHAM Limited].

1.4.6 Finite Element and Boundary Element Based Free Surface Flow Methods

Finite Element

Dhatt et al (1990) developed a two dimensional finite element model to simulate mould filling with heat transfer using a VOF type tracking method. The finite element grid is fixed and the advection of the molten metal is captured by a *pseudo-concentration function F* or the volume fraction as in the VOF method. A *F*-smoothing technique was applied to reduce the smearing at the interface. And a Taylor-Galerkin method was used to model heat transfer during filling.

Lewis et al (1991a, 1991b) [and Usmani et al (1992)] have developed a finite element code to simulate the fluid flow and heat transfer in mould filling problems. The VOF method was adopted to track the liquid front and the Taylor-Galerkin method was used to solve for the Navier-Stoke equation and energy transfer.

Codina et al (1992) developed a finite element model based on the pseudo-concentration technique (ie. a VOF type method) to simulate mould filling processes. The main features of their model is the use of the *Streamline Upwind/Petro-Galerkin* (SUPG) formulation to deal with convective flow, div-stable velocity-pressure interpolations with discontinuous pressures, the elimination of the pressure via an iterative penalty formulation, and the temporal integration using a trapezoidal rule.

Shen (1992) of Cornell University developed a finite element model to simulate non-Newtonian non-isothermal Hele-Shaw filling problems.

Boundary Element

Medina et al (1991) developed a boundary element method to model free surface flow. A *consistent formulation* was introduced to deal with the non-linearities which exist in free surface boundaries. The flow field equation is integrated and the free boundary conditions are applied on the unknown geometry by means of series expansions.

1.4.7 Remarks on the Literatures Survey

The author draws the following conclusions after carrying out the literature search on the free surface flow techniques:

(1) There does not seem to be a great difference whether the modelling of free surface flow is done in either FE or CV formulation, especially in the way the interface is tracked. A pseudo-concentration function or the equivalent volume-of-fluid fraction is being adopted in both frameworks to deal with the tracking of the interface. The attractions of using FE formulation are primarily with the flexibility of modelling arbitrary shapes and the ease to incorporate stress analysis. However, with the realisation of control-volume unstructured mesh (CV-UM) or irregular control-volume (ICV) methods (such as UIFS [Chow (1993)] and ASTEC [AEA Technology]), this flexibility of modelling arbitrary shapes is also available to CV based formulation. In the case of ASTEC, the capability of modelling free surface flows using control-volume unstructured meshes is already available (see *Section 1.1.2*). Also, stress analysis is included in UIFS which is an in-house code developed at the University of Greenwich. Furthermore, as mentioned in Section 1.1.2, the study of mould filling forms part of the research programme in casting technology based on CV formulation at Greenwich. Therefore, it is more appropriate for the author to develop the filling module in CV formulation. So that it can be implemented in other CV based CFD codes, whether they use structured or unstructured mesh, in the future.

(2) It emerges that explicit VOF type volume-tracking methods are most widely used by researchers to model mould filling applications, even though the resolution of the results are not as good as those obtained by front tracking methods. This is based on the fact that VOF type methods occupy less storage space and has less complication in the programming logic than front tracking methods, for example the particle tracking method. And yet it provides sufficient all round front tracking capability. This is important if other physics, such as heat transfer and solidification, are to be included in the same code.

(3) An important issue when deciding which tracking method to use is how and what technique does the prospective method adopt to handle the problem of numerical smearing at the interface. In the case of VOF type methods, the advection/differencing schemes chosen to preserve the gas-liquid discontinuity at the interface are most important. When using this kind of differencing scheme, small numerical errors in the approximation are always expected. To minimise this type of errors, higher order differencing schemes may be used at the expense of higher CPU time. Therefore, the researcher must weight the balance between the requirement of the applications and the resource available when choosing a differencing scheme.

(4) The SEA algorithm is chosen for this study based upon two grounds:

(i) The requirements of modelling the entrainment of impurities and air voids in the casting during filling, points to the need of simulating the both flow fields for the air and liquid phases. In the basic VOF method, additional calculation is required to model the effects due to the air phase.

(ii) Since the SEA algorithm is a VOF type method, the storage overheads for tracking the interface are minimal. The SEA algorithm only requires a scalar variable to be stored in each cell to indicate whether a cell is full, half-filled or empty of liquid. Hence, additional dependent variables required to facilitate additional physics such as heat transfer and solidification can be accommodated. Furthermore the CPU time required by the van Leer scheme is small in comparisons with other higher order differencing scheme [Leonard (1988), Bull(1990)].

1.5 Outline of the Thesis

The results of the present study is organised into seven chapters.

The first chapter which we have just seen, aims to give a broad introduction of industrial applications involving free surface flows, and focuses on its application in mould filling which has profound influence on casting processes. In addition, a comprehensive literature survey is included in this chapter to give the reader an up-to-date view of current research activities on the subject of interest.

A concise account of the derivation of equations which govern transport phenomena including the conservation of mass, momentum and energy, is presented in *Chapter 2*. Furthermore, the numerical solution procedure of solving these governing equations in the context of a control volume formulation is illustrated in the chapter.

The Scalar Equation Algorithm (SEA) exploited in this study originates from a report by Liu Jun (1986a) who developed the method in two-dimensions. The theory, assumptions, auxiliary information, and detailed derivation of the SEA method extended to three dimensions by the author for mould filling applications, are outlined in *Chapter 3*.

In *Chapter 4*, the results generated by the SEA method, in both two and three dimensions are presented. These are validated against experimental data to demonstrate the capability of the method in capturing waves and air entrainment.

Chapter 5 describes the potential impact of parallel computing on the computational efficiency of CFD simulation software when solving complex industrial problems. The implementation of the SEA method into parallel Harwell-FLOW3D and the results on the speed-up gain of the parallel version over the serial are also described.

Chapter 6 is devoted to the development of a novel technique to model heat transfer between two or more immiscible fluids. This technique forms the basis for the integration or coupling of free surface flow with a solidification model, in order to simulate filling with phase change simultaneously. The solidification model used in this study is based on the one proposed by Voller and Prakash (1987b).

Finally, the conclusion of the study and suggestions for future work are presented in *Chapter 7*, followed by references and an appendix.

Chapter 2 *BASIC MATHEMATICAL EQUATIONS FOR FLUID FLOW*

2.1 Introduction

The purpose of this Chapter is to outline a general framework of how to develop a mathematical model of free surface fluid flow from physical laws. The model is then implemented as an attachment to two commercial Computational-Fluid-Dynamics (CFD) codes namely, PHOENICS [CHAM Limited] and Harwell-FLOW3D [AEA Technology]. The details of implementing the model attached to both PHOENICS and Harwell-FLOW3D as a stand-alone module, can be found in *Chapters 3* and *5* respectively.

This Chapter is organised into the following sections:

- *Section 2.2* describes the governing equations of fluid dynamics problem under study and the associated auxiliary relations.
- *Section 2.3* describes the control-volume (CV) formulation used to discretise these governing equations into algebraic finite-difference equations so that they can be solved numerically.
- *Section 2.4* is focused on the general numerical solution procedure employed to solve these algebraic equations in CFD codes.
- *Section 2.5* is the summary of this Chapter.

2.2 Governing Equations

Mathematics has been used by scientists to describe physical phenomena since ancient time. The basis of CFD techniques is built upon well-known physical laws, which govern all physical phenomena that occur in everyday life. These laws include the conservation of mass, energy and momentum. They are usually expressed in terms of partial differential equations (PDEs). These PDEs can rarely be solved by analytical means since most physical phenomena are non-linear in nature. However, this non-linearity is particularly well posed to be solved by iterative numerical methods, running on electronic digital computers.

For example, mould filling is a physical phenomenon which involves fluid flow and heat transfer and so it is governed by conservation principles of mass, momentum, and energy.

2.2.1 Coordinate system

Before describing the governing equations in PDEs, it is necessary to specify the coordinate system used throughout the study. Simulations of free-surface flow are presented in later chapters in cartesian coordinates in either two dimensions (2-D) or three dimensions (3-D). *Figure 2.1* illustrates a typical cartesian coordinate system (x,y,z). The mathematical model will be developed in cartesian coordinates. Similar free-surface flow models have been developed in both polar-cylindrical and cartesian coordinate systems in the past [Maxwell (1977), Liu Jun (1986a, 1986b)]. Future possible extension to the model could be to incorporate polar-cylindrical coordinate system.

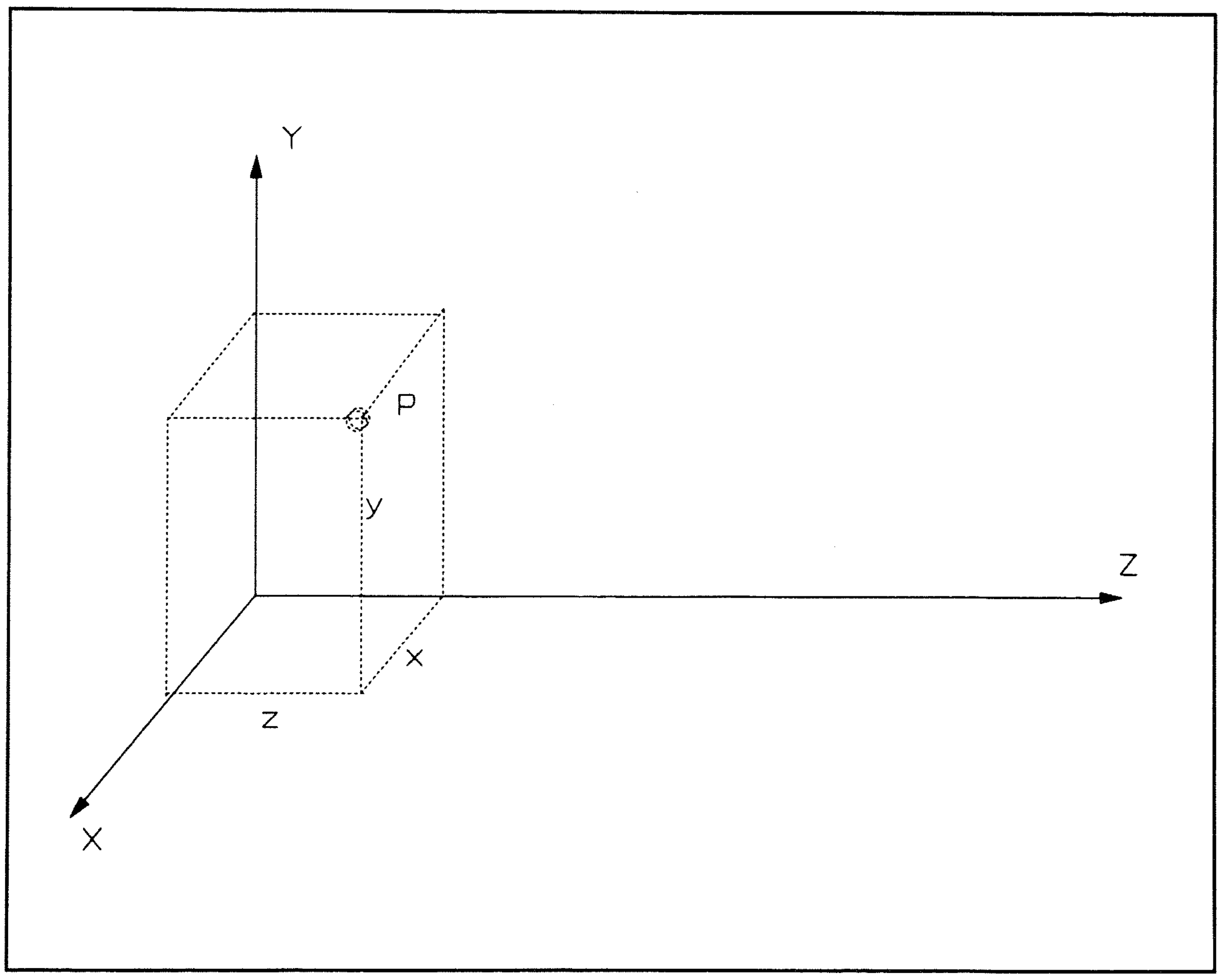


Figure 2.1 Cartesian coordinates (x, y, z)

2.2.2 The Mass Conservation Equation

The mass conservation principle states that, in the context of a differential control volume, that *the net rate at which mass enters the control volume (ie. inflow - outflow) must equal to the rate of increase of mass stored within the control volume.* The mass conservation equation may be expressed in vector form as:

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho \underline{u}) = S \qquad (2.1)$$

where t is the time, ρ is the fluid density, $\underline{u}$ is the resultant velocity vector in the three space dimensions (x,y,z), and ∇.( ) is the divergence operator on a vector field. The three terms in *Equation 2.1*, from left to right, are the transient, convection and S which is the *source* or *sink* term of mass.

2.2.2.1 The Mass Continuity Equation

Further, the mass conservation principle is based upon the physical principle that matter could neither be created nor destroyed. As a consequence, a particular case of *Equation 2.1* is the mass continuity equation (*Equation 2.2*) where S becomes zero. Here, the mass continuity equation merely states that the mass inflow (due to rate of change of mass and convection) to a specific control-volume must equal to its mass outflow. Therefore, the net outflow or inflow is zero.

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho \underline{u}) = 0 \tag{2.2}$$

From fluid mechanics, the mass continuity equation has an important significance with regard to fluid flow. The density ρ and the velocity field $\underline{u}$ derived from the momentum equation (*Equation 2.3*) satisfy the mass continuity equation (*Equation 2.2*). This is due to the pressure field in the momentum equation which is coupled indirectly with the mass continuity equation in the anticipation of the so-called pressure-correction formula (*Equation 2.42*). That is, the pressure field must be adjusted so that the velocity field satisfies the mass continuity equation. So without this constraint, momentum will not be conserved.

2.2.3 The Conservation of Momentum Equation

The conservation of momentum equation for fluid flow, in vector form, is given as:

$$\frac{\partial \rho \underline{u}}{\partial t} + \nabla.(\rho \underline{u}\underline{u}) = \mu \nabla^2 \underline{u} - \nabla p + \rho g + S_{ext} \tag{2.3}$$

where μ is the dynamic viscosity of fluid, p is the pressure, g is the acceleration under gravity, S_{ext} denotes the external source, such as natural convection due to thermal expansion; ∇( ) is the gradient operator on a scalar field, and ∇^2 is the Laplacian operator on a vector field. The terms from left to right in *Equation 2.3* are the transient, convection, diffusion, pressure gradient, body force due to gravity and external momentum source. With constant viscosity and constant density, the momentum equation (*Equation 2.3*) becomes the well-known Navier-Stokes equation.

2.2.4 The Conservation of Energy Equation

Let h stand for the enthalpy per unit mass. Then the conservation of energy equation for a specific control volume is given by:

$$\frac{\partial(\rho h)}{\partial t} + \nabla.(\rho \underline{u} h) = k\nabla^2\left(\frac{h}{C_p}\right) + S_h \qquad (2.4)$$

where k is the thermal conductivity of the fluid, C_p is the specific heat capacity and S_h is the source/sink term of enthalpy.

2.2.5 The General Partial Differential Equation

By observing *Equations 2.1* to *2.4*, it is not difficult to appreciate that all these conservation equations can be generalised and expressed in a common form. The generalised conservation equation, according to the notations used in Patankar (1980) is shown below:

$$\frac{\partial(\rho \Phi)}{\partial t} + \nabla.(\rho \underline{u}\Phi) = \nabla.(\Gamma_\Phi \nabla\Phi) + S_\Phi \qquad (2.5)$$

where Φ represents the dependent variable to be solved for, such as temperature (T), enthalpy (h) and concentration (C), ρ denotes the fluid density and $\underline{u}$ is the resultant velocity vector of the three velocity components. Γ_Φ is the exchange or diffusion coefficient representing the general fluid properties such as viscosity (μ) or thermal conductivity (k).

Equation 2.5 consists of four terms, from left to right, the *transient* term which is considered as the rate of change of Φ *per unit volume*. The second term is the *convection* term which denotes the convective flux $\rho\underline{u}\Phi$ carried by the general flow field $\rho\underline{u}$. The third term is the *diffusion* term which represents diffusive flux caused by the gradient of Φ. And the last is the *source* term where additional terms can be added. The equations for the various Φ's are distinguished through the source terms. For example if Φ is the velocity then S_Φ would contain the pressure gradient, body force, and other viscous terms which have not been included in the nominal diffusion term. Note that the diffusion term in *Equation 2.5* is expressed in a different form to the one in *Equation 2.3*.

2.2.6 Auxiliary Relations

Although the governing equations have been described above, the characteristics of a particular mathematical model have not yet been established, and hence the model is not complete. To do so, it requires the following auxiliary information which includes:

- Boundary conditions
- Initial conditions
- Material properties
- Other physical relations.

Boundary Conditions

The most influential auxiliary relations to a mathematical model are probably the boundary conditions. They are assigned at the edges of the calculation domain and internal blockages. The effects of the boundary conditions are *transported* into the domain when solving the relevant conservation equations during simulations. Sometimes the conservation equations may also be referred to as the *transport equations*. In case of mould filling, the physical boundaries that are most likely to be encountered would be the inlet, outlet, impervious walls, internal plate or blockages, surface tension and planes of symmetry. These boundary conditions are usually known to the user in advance. Some of the boundary conditions and the variables involved are listed below:

<u>*Inlet*</u>

Velocity:	Velocity components are specified at the cell centres of the inlet. Since volumetric conservation is used in the case of filling, a volumetric source is needed at the inlet cells to specify the incoming volume flux.
Pressure:	No pressure value specified.
Temperature:	Fixed temperature or heat flux.

Outlet

Velocity: Volume fluxes leaving the solution domain at the outlet cell faces (downstream) are evaluated using the velocity component at the upstream cell faces.

Pressure: For the outflow, a zero pressure boundary is usually set.

Temperature: Heat flux is evaluated using temperature at cell centre.

Wall

Velocity: No slip boundary condition apply. Zero mass or volume flux. The values of the velocity components parallel to the cell face can be set.

Pressure: Not specified.

Temperature: Fixed temperature or heat flux.

Symmetry

Velocity: The velocity component normal to the symmetry plane is forced to zero, and those parallel to the symmetry plane have the cell centre values.

Pressure: The pressure at the cell faces have the values of the cell centres.

Temperature: No heat flux can be transferred across the symmetry plane. The cell faces have the same temperature values as the cell centres.

Blockages

To make internal walls or block off certain parts of the solution domain by making them inaccessible to the fluid flow field calculation in PHOENICS, a porosity factor can be used. A zero porosity value in a cell means that it is blocked. The surfaces of a blocked cells have no slip wall boundary conditions activated automatically.

Velocity: Zero velocity specified at velocity components normal to the blocked cells. Zero mass or volume flux. No slip boundary condition applies to all blocked cell faces.

Pressure: No pressure specified.

Temperature: Temperature and heat flux can be specified.

Initial Conditions

This kind of condition refers to values assigned to the dependent variable(s) at the beginning of the computation. In the case of mould filling, the dependent variables are the velocity components, pressure, temperature and the initial position of the air-liquid interface. If appropriate initial values are specified, even for steady-state problems the solution should converge faster, otherwise the solution may either converge slower or even diverge. In time-dependent simulations the initial conditions specify the old-time boundary condition of the problem.

Material Properties

There are two areas concerning the material properties of the medium or media used in the simulation which have considerable influence to the accuracy of the prediction.

The first area of concern is that of the availability of accurate material property values. Without accurate data for the material under simulation, it is impossible for the model to give plausible predictions, even though the model has all the appropriate physical capabilities incorporated into it. For example, in solidification modelling, a slight inaccuracy in the solidus and liquidus temperatures of the material undergoing simulation may yield completely unrealistic results.

The second area of concern is that of the physical relations or formulations used to determine the material properties during computation, if more than one phase or material is to be simulated. These relations may be derived from theory, experiments or simply by a heuristic rule (as used in ASTEC's [AEA Technology] filling module). For instance, in the case of mould filling, the mixture density of liquid and gas within a cell is determined by a linear approximation relationship which involves a volume fraction.

Other Physical Relations

As mentioned in *Section 2.2.5*, additional terms of a dependent variable Φ which cannot be included in either transient, convection or diffusion term are treated as source terms. These additional terms are essentially the physical relations which give the unique characteristics to a model. Here are a few examples:

1. Body forces such as gravity, pressure gradient and centrifugal forces added to the momentum equation will affect the behaviour of fluid flow.
2. The volume source term which specifies the amount of volume *flow* into the computational domain when GALA [Spalding (1974, 1977)] is used. See *Section 3.3.2* of *Chapter 3* for details.
3. The latent heat source term in an enthalpy based solidification model [Voller et al (1985, 1987a)] which governs the phase-change of the fluid.

2.3 Numerical Discretisation of Governing Equations

The numerical integration scheme used to solve the set of continuous and generally non-linear governing differential equations as described in *Section 2.2* is presented in the following sections.

2.3.1 Introduction to Control-Volume Formulation

In order to solve continuous PDEs using computational methods, it is necessary to transform these PDEs into algebraic finite-difference equations by applying discretisation. A number of discretisation schemes are available for this purpose, such as finite-difference (FD), control-volume (CV) or finite-volume (FV), finite-element (FE) and boundary-element (BE) formulations. However, the most widely use formulation for solving fluid flow problems is that of control-volume. The CV formulation is based upon the divergence theorem by Gauss as shown in *Equation 2.6.*

$$\iiint_V \nabla.\underline{F}\, dV = \iint_S \underline{F}.\underline{n}\, dS \tag{2.6}$$

where V denotes the finite control volume, S denotes the closed surface around the volume, $\underline{F}$ denotes the vector field defined on V, and $\underline{n}$ is the unit outer normal to S.

The divergence theorem expresses a relationship between a volume integral extended over a computational domain and a surface integral over the boundary of the domain. This enables the computational domain to be divided into a number of sub-domains or sub-control-volumes, such that there is one control-volume surrounding a grid point.

The main attraction of divergence theorem is that, it ensures the integral conservation of matter, such as mass over the whole computational domain. This characteristic remains valid for any number of grid points. A vital implication of this feature of CV formulation is that no matter how coarse or fine a mesh (or grid) is being specified by the user, the conservation of the dependent variable is always maintained. However, when finite-element formulation is used, the conservation of dependent variable at individual element is not necessarily conserved.

2.3.2 The Finite Difference Grid

For structured CV codes, such as PHOENICS and Harwell-FLOW3D, a typical structured mesh is shown in *Figure 2.2*. The mesh represents the physical domain which is sub-divided into a number of cartesian control-volumes or cells which appear as orthogonal cuboids in three-dimensions (*Figure 2.3*), and quadrilateral in two-dimensions (*Figure 2.4*).

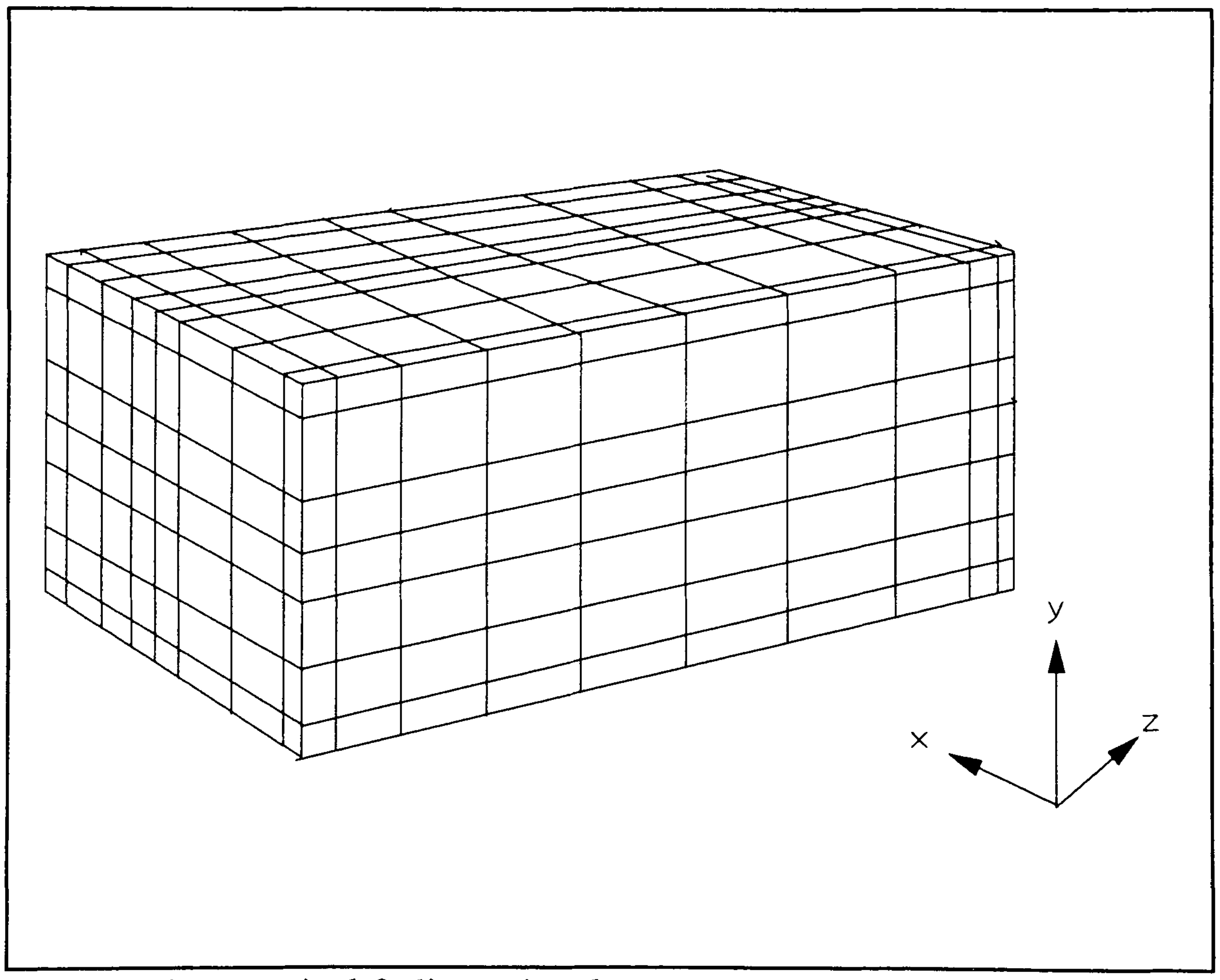


Figure 2.2 A typical 3-dimensional structured mesh

These control-volumes are non-overlapping. The mesh needs not be uniformly spaced. More cells with smaller spacing can be placed in areas where rapid changes in the

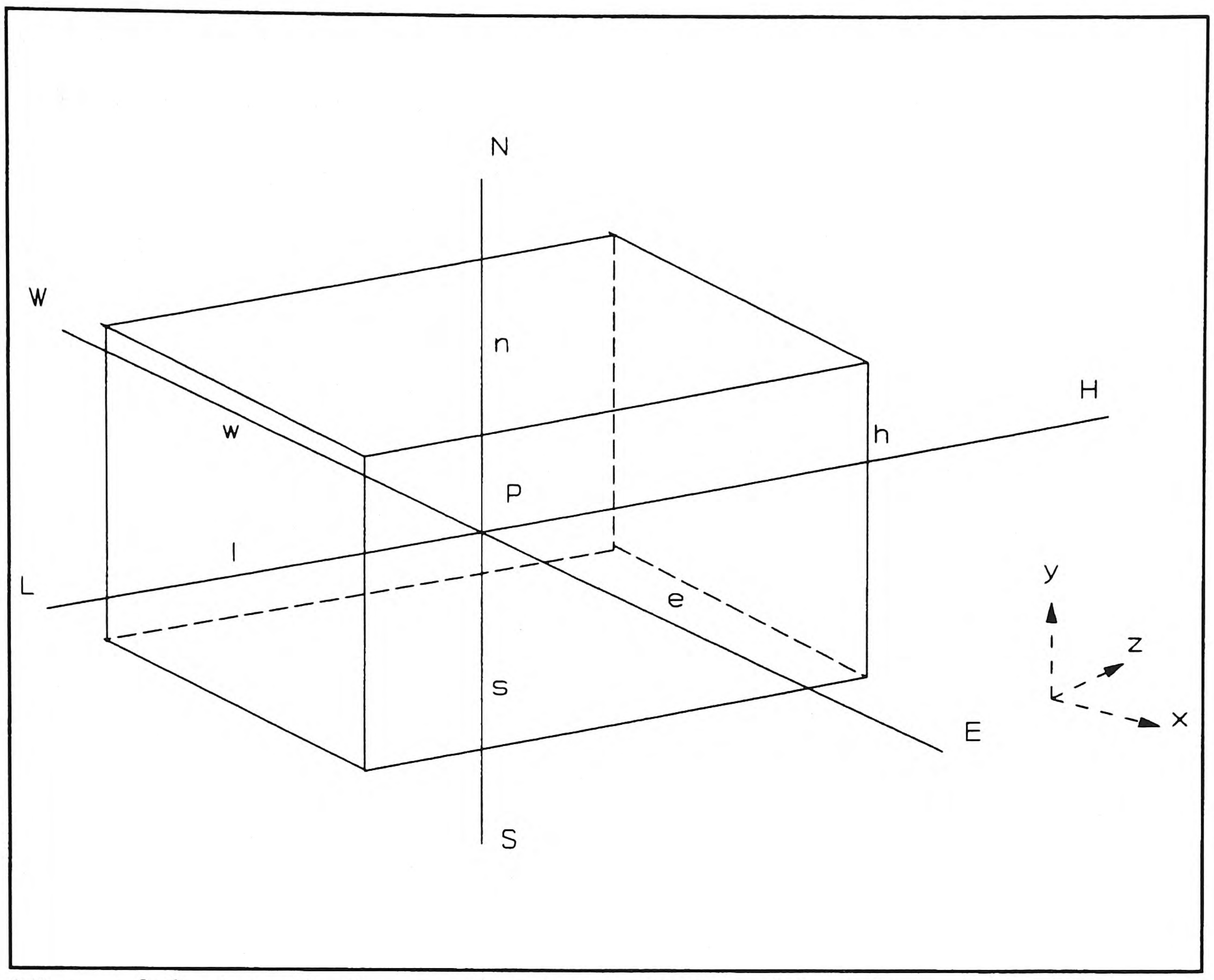


Figure 2.3 A typical control-volume (CV) cell in 3-D

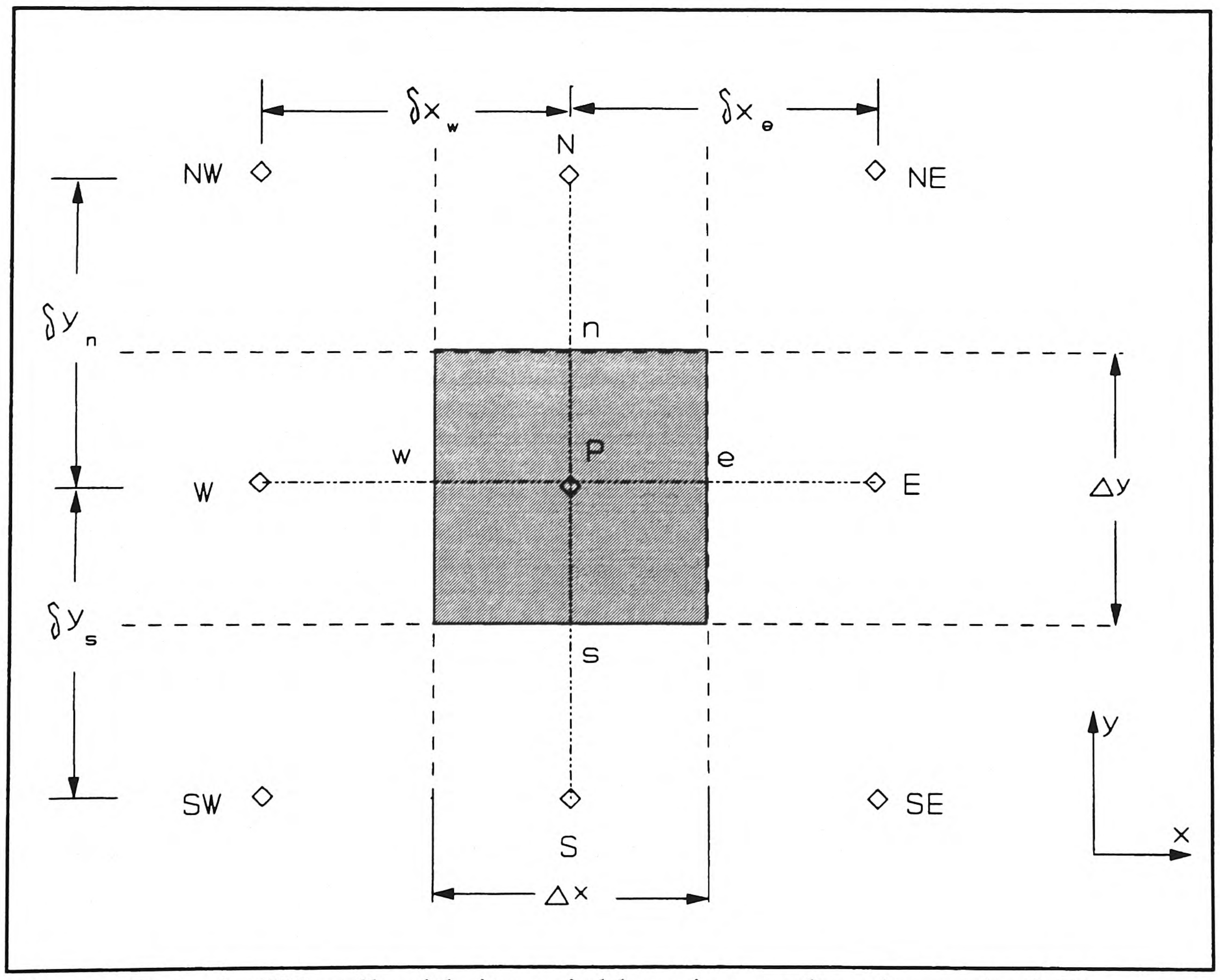


Figure 2.4 A CV cell with its neighbouring nodes shown in 2-D

gradient of flow are anticipated; for example, near walls, inlet and sudden expanded regions. Hence, more information from the area of interest can be extracted. On the other hand, larger and fewer cells may be used in areas where very little changes in the flow field are expected.

Each cell of the domain has a grid point or node at its geometric centre. The node point P, as shown in *Figure 2.3*, is surrounded by six neighbouring nodes. They are the East and West in the x-direction (denoted by E and W); the North and South in the y-direction (denoted by N and S); and the High and Low in the z-direction (denoted by H and L). The cell face locations where the grid lines joining P with its neighbours intersect the six cell faces are marked as e, w, n, s, h and l respectively. (In two-dimensions, the node point P will have four neighbours and four cell face locations as shown in *Figure 2.4*). The value stored at the node P is assumed to prevail over the entire control-volume; whereas the value stored at a cell face location is assumed to prevail over the whole area of that cell face only.

Scalar dependent variables, such as temperature and concentration are stored at the grid node (P), whereas velocity components (u, v, and w) are *usually* stored at the cell face locations. This is known as the *staggered* grid arrangement; see *Section 2.3.2.1* below. This kind of grid arrangement is widely adopted by most structured CV codes developed in late 1970's and early 1980's, for example PHOENICS. In recent years, commercial CFD codes with *non-staggered* grid arrangement (in which velocity components are stored in the same way as other variables at the main grid node) have emerged; see *Section 2.3.2.2*. Harwell-FLOW3D and ASTEC [AEA Technology] are two example codes; whereby the former is structured and the latter is unstructured. The success of their implementation is largely due to a special interpolation algorithm for velocity components developed by Rhie and Chow (1982).

From now on, a staggered grid is assumed to be used in the discretisation of momentum equation (*Equation 2.3*) throughout this thesis, except where it states otherwise.

2.3.2.1 Staggered Grid

The staggered grid arrangement for the velocity component first appeared in the MAC method by Harlow and Welch (1965). The reason for its deployment was to suppress *checkerboard effect* (as described by Patankar (1980)) or spurious pressure oscillation, due to the coupling of the momentum and continuity equations when velocity is stored in a non-staggered grid. A well documented account for checkerboard effect can be found in Patankar (1980) pages 113 to 120.

Velocity components u and v denoted by arrows, are stored in a staggered grid arrangement on x-y plane in two-dimensions as shown in *Figure 2.5*. The small circles denote the main grid nodes where the scalar variables are stored. The velocity components are located at the main cell faces between the main grid nodes, in both x and y directions. In effect, every velocity component has its own control-volume which can be defined by its four neighbouring arrows that are normal to itself. The typical control-volumes for both u- and v- velocities are shown as shaded regions. Considering the control-volumes of the main nodes with that of the velocities, it is not hard to see that the velocity control-volumes are over-lapped or staggered by half-a-cell, ahead of their associated main control-volumes. Also, it is not difficult to extend the staggered grid to three-dimensions to include the w- velocity in the z-direction.

The important features of staggered grid can be summarised as follows:

1. When discretising the continuity equation (*Equation 2.2*) for a typical control-volume cell P, the equation would contain the differences of velocity components at *adjacent* nodes (eg. u_e and u_w). They are stored at the cell faces of P as shown in *Figure 2.5*. This will give a reasonable pressure field for the momentum equation and will also satisfy the continuity equation. Whereas with the non-staggered grid arrangement (shown in *Figure 2.6*), the continuity equation would contain the differences of velocity components at *alternate* main grid nodes, such as u_W and u_E. This would result in the continuity equation taking a spurious, oscillating pressure field as a possible solution.
2. The pressure difference between two adjacent main grid nodes becomes the natural driving force for the velocity node located in between.

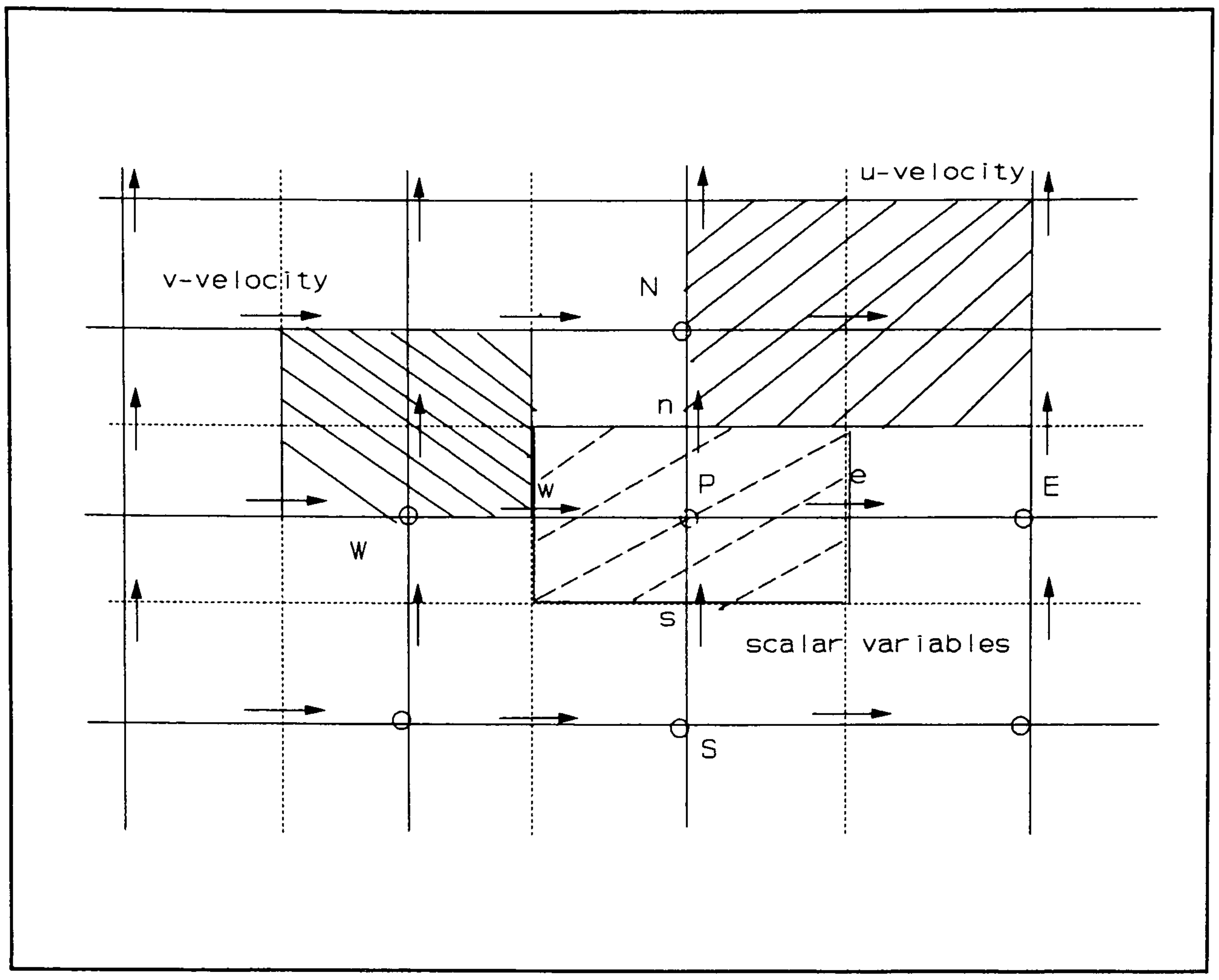


Figure 2.5 Staggered grid for u and v velocity components

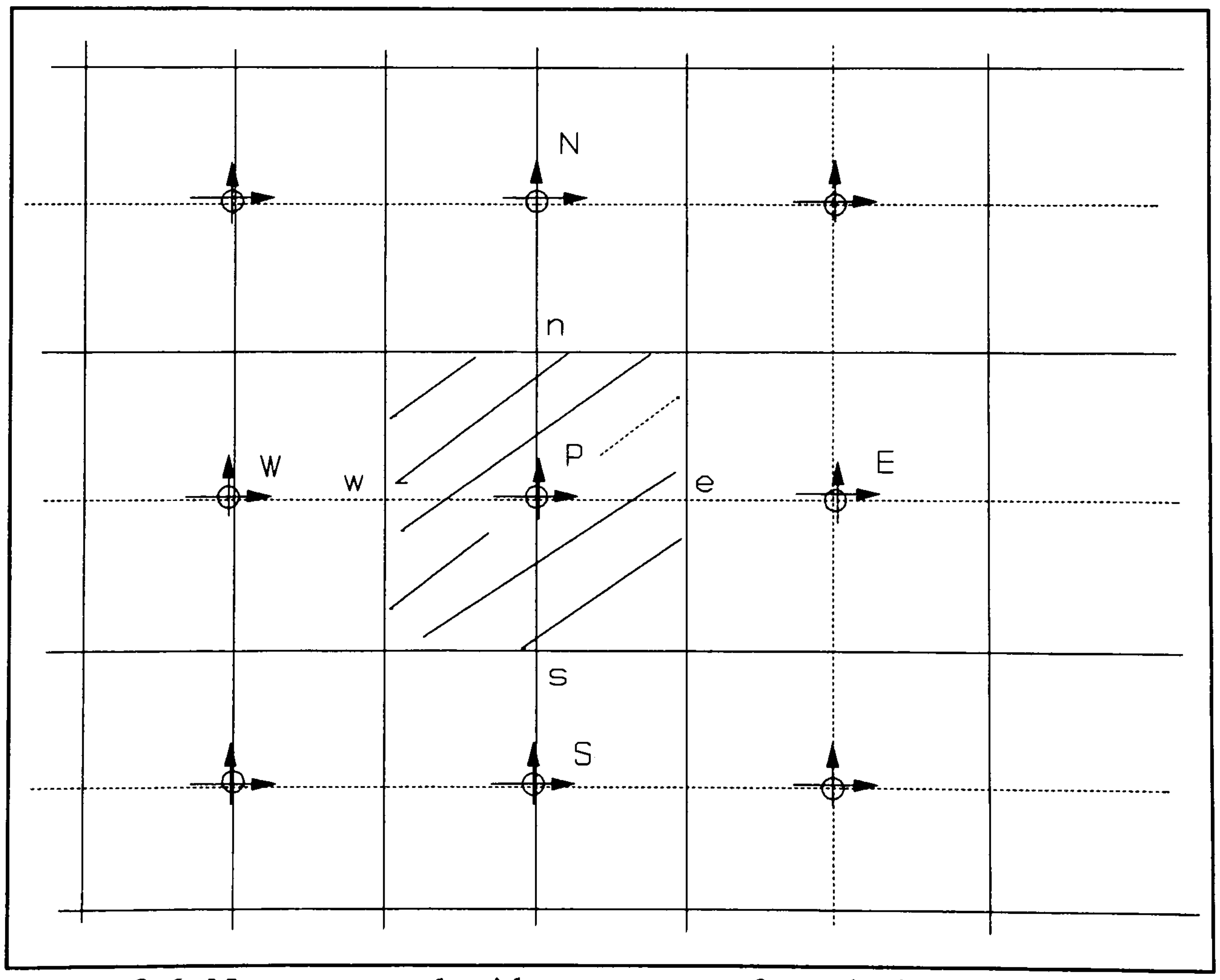


Figure 2.6 Non-staggered grid arrangement for velocity components

3. There is no need to interpolate the value of velocity components at the main grid cell faces, since they are directly available.

A drawback of staggered grid arrangement is that it requires to use three separate grids for a two-dimensional problem and four grids for three-dimensions. This has implications on the amount of hardware storage that it would require; and adding complexity to the house-keeping of geometric and indexing information in programming. The latter implication has proved to be a serious disadvantage for staggered grid arrangement to be used in unstructured codes, such as ASTEC and UIFS at University of Greenwich [Chow (1993)]. Hence, ASTEC, Harwell-FLOW3D and UIFS adopted the non-staggered grid approach for this reason.

2.3.2.2 Non-staggered Grid

As mentioned in the previous section the non-staggered grid, as shown in *Figure 2.6*, has an inherent problem with the stability of the pressure field when solving the momentum equation. The Rhie-Chow interpolation algorithm [Rhie and Chow (1982)] *suppresses* the pressure oscillation by giving realistic approximation of velocity values at cell face locations. For unstructured codes, many difficulties involving node connectivity within unstructured cells, such as a mixture of tetrahedra and cuboids, have been avoided by using the Rhie-Chow interpolation algorithm. This is because the programming logic required to set up a staggered grid for approximating momentum fluxes at cell faces, for a mixture of tetrahedra and cuboids, will be very complicated if not impossible.

A full account of the advantages and disadvantages concerning the staggered and non-staggered grids can be found in two papers by Melaaen (1992a, 1992b).

2.3.3 Discretisation of the General Conservation Equation

To obtain the finite-difference equations in algebraic form, the general conservation equation (*Equation 2.5*) for a general variable Φ is integrated over a finite volume δV and over a finite increment of time δt. Replace $\underline{F}$ in the divergence theorem (*Equation 2.6*) by $(\rho \underline{u} \Phi - \Gamma_\Phi \nabla \Phi)$, to obtain *Equation 2.7*.

$$\iiint_V \nabla.(\rho\, \underline{u}\Phi - \Gamma_\Phi \nabla\Phi)\, dV = \iint_S (\rho\, \underline{u}\Phi - \Gamma_\Phi \nabla\Phi).\underline{n}\, dS \tag{2.7}$$

Apply *Equation 2.7* to the general conservation equation (*Equation 2.5*); integrating with respect to time and expanding the results gives:

$$\begin{aligned}\frac{\partial}{\partial t}\iiint_V \rho\Phi\, dV &+ \iint \rho\underline{u}\Phi\big|_x^{x+\Delta x} dydz - \iint \Gamma_\Phi\nabla\Phi\big|_x^{x+\Delta x} dydz \\ &+ \iint \rho\underline{u}\Phi\big|_y^{y+\Delta y} dxdz - \iint \Gamma_\Phi\nabla\Phi\big|_y^{y+\Delta y} dxdz \\ &+ \iint \rho\underline{u}\Phi\big|_z^{z+\Delta z} dxdy - \iint \Gamma_\Phi\nabla\Phi\big|_z^{z+\Delta z} dxdy \\ &= \iiint_V S_P\, dV\end{aligned} \tag{2.8}$$

After taking the limits, *Equation 2.8* becomes:

$$\begin{aligned}&\frac{[\,(\rho\Phi)^{new} - (\rho\Phi)^{old}\,]_P}{\delta t}\, V \\ &+ [\rho\underline{u}\Phi - \Gamma_\Phi\frac{\partial\Phi}{\partial x}\,]_e\, A_e - [\rho\underline{u}\Phi - \Gamma_\Phi\frac{\partial\Phi}{\partial x}\,]_w\, A_w \\ &+ [\rho\underline{u}\Phi - \Gamma_\Phi\frac{\partial\Phi}{\partial y}\,]_n\, A_n - [\rho\underline{u}\Phi - \Gamma_\Phi\frac{\partial\Phi}{\partial y}\,]_s\, A_s \\ &+ [\rho\underline{u}\Phi - \Gamma_\Phi\frac{\partial\Phi}{\partial z}\,]_h\, A_h - [\rho\underline{u}\Phi - \Gamma_\Phi\frac{\partial\Phi}{\partial z}\,]_l\, A_l \\ &= V_\Phi S_{\Phi P}\end{aligned} \tag{2.9}$$

where $(\ )^{new}$ and $(\ )^{old}$ denote both the fluxes at the current and previous time step respectively. All other terms in *Equation 2.9* can be taken at the current time step, if an *implicit* formulation is required. On the other hand, an *explicit* formulation can be formed by taking all these terms at the previous time step. For the reason of numerical stability implicit formulation is usually used. Also, the *A*'s represent the areas of various cell-faces.

Now approximate the partial derivatives of diffusion terms using backward Euler method by assuming a piecewise-linear profile between two adjacent grid points. This gives the following:

$$\begin{aligned}
&\frac{[\,(\rho\Phi)^{new} - (\rho\Phi)^{old}\,]_P}{\delta t} V \\
&+ (\rho\underline{u}\Phi)_e\, A_e - \Gamma_\Phi \frac{(\Phi_E - \Phi_P)}{\delta x_e} A_e - (\rho\underline{u}\Phi)_w\, A_w + \Gamma_\Phi \frac{(\Phi_P - \Phi_W)}{\delta x_w} A_w \\
&+ (\rho\underline{u}\Phi)_n\, A_n - \Gamma_\Phi \frac{(\Phi_N - \Phi_P)}{\delta y_n} A_n - (\rho\underline{u}\Phi)_s\, A_s + \Gamma_\Phi \frac{(\Phi_P - \Phi_S)}{\delta y_s} A_s \\
&+ (\rho\underline{u}\Phi)_h\, A_h - \Gamma_\Phi \frac{(\Phi_H - \Phi_P)}{\delta z_h} A_h - (\rho\underline{u}\Phi)_l\, A_l + \Gamma_\Phi \frac{(\Phi_P - \Phi_L)}{\delta z_l} A_l \\
&= V_\Phi S_{\Phi P}
\end{aligned} \tag{2.10}$$

Apply the substitution,

$$C \equiv \rho\underline{u}A, \qquad D \equiv \Gamma_\Phi A/\delta i$$

where C represents the mass flux through convection and D represents the diffusion flux across the cell faces. The distance between adjacent grid point is represented by δi, where i indicates the three space directions x, y and z respectively. D always remains positive whereas C can either be positive or negative depending on the direction of fluid flow [Patankar (1980)].

A simplified *Equation 2.10* becomes:

$$\begin{aligned}
&\frac{[\,(\rho\Phi)^{new} - (\rho\Phi)^{old}\,]_P}{\delta t} V \\
&+ C_e\Phi_e - C_w\Phi_w + C_n\Phi_n - C_s\Phi_s + C_h\Phi_h - C_l\Phi_l \\
&- D_e(\Phi_E - \Phi_P) + D_w(\Phi_P - \Phi_W) - D_n(\Phi_N - \Phi_P) \\
&+ D_s(\Phi_P - \Phi_S) - D_h(\Phi_H - \Phi_P) + D_l(\Phi_P - \Phi_L) \\
&= V_\Phi S_{\Phi P}
\end{aligned} \tag{2.11}$$

However, the convection terms of *Equation 2.11* cannot be approximated in a similar manner as the diffusion terms. The central difference approximation of the convection terms is only valid if the Peclet[1] number is not exceeding 2, else the solution may become unrealistic and oscillatory. So to avoid this numerical diffusion problem, many differencing schemes have been devised in the past for example, Upwind [Courant et al (1952)], Hybrid [Spalding (1972)], Quick [Leonard (1980)], and Van Leer [Van Leer (1977)]. Among all these schemes the simplest is the upwind which is also widely available in many commercial CFD codes. The upwind scheme assumes that *the values of Φ at an interface is equal to the value of Φ at the grid point on the 'upwind' side of the face* [Patankar (1980)]. For example the convection flux across the east face of cell P is

$$\Phi_e = \Phi_P \quad \text{if } C_e > 0$$

and $$\Phi_e = \Phi_E \quad \text{if } C_e < 0 \tag{2.12}$$

Similar constraints apply to the convection flux at the west face of the cell.

The cell face values of Φ of the convection terms in *Equation 2.11* can now be written as:

$$\begin{aligned}
C_e\,\Phi_e &= \Phi_P [\![C_e, 0]\!] - \Phi_E [\![-C_e, 0]\!] \\
C_w\,\Phi_w &= \Phi_W [\![C_w, 0]\!] - \Phi_P [\![-C_w, 0]\!] \\
C_n\,\Phi_n &= \Phi_P [\![C_n, 0]\!] - \Phi_N [\![-C_n, 0]\!] \\
C_s\,\Phi_s &= \Phi_S [\![C_s, 0]\!] - \Phi_P [\![-C_s, 0]\!] \\
C_h\,\Phi_h &= \Phi_P [\![C_h, 0]\!] - \Phi_H [\![-C_h, 0]\!] \\
C_l\,\Phi_l &= \Phi_L [\![C_l, 0]\!] - \Phi_P [\![-C_l, 0]\!]
\end{aligned} \tag{2.13}$$

where $[\![\ ]\!]$ represents a maximum function which returns the greater of the two arguments.

The source term is given as:

[1]Peclet number is defined as $Pe \equiv \rho \underline{u} \delta i \,/\, \Gamma$, ie the strength between convection and diffusion.

$$\iiint_V S_\Phi \, dV = S_\Phi V_P \qquad (2.14)$$

So often the source term is not a constant but a function of Φ; therefore it is necessary to linearise the source term so that the resulting finite difference equation remains linear. The source term can be expressed in a general form as below [Patankar (1980)]:

$$S_\Phi = S_C + S_P \Phi_P \qquad (2.15)$$

Substituting *Equations 2.13* and *2.15* into *2.11* gives the discretised algebraic form of the general differential *Equation 2.8.*

$$a_P \Phi_P^{new} = a_E \Phi_E + a_W \Phi_W + a_N \Phi_N + a_S \Phi_S + a_H \Phi_H + a_L \Phi_L + b \qquad (2.16)$$

where

$$a_E = D_e + [\![-C_e, 0]\!]$$

$$a_W = D_w + [\![C_w, 0]\!]$$

$$a_N = D_n + [\![-C_n, 0]\!]$$

$$a_S = D_s + [\![C_s, 0]\!]$$

$$a_H = D_h + [\![-C_h, 0]\!]$$

$$a_L = D_l + [\![C_l, 0]\!]$$

$$a_P^{old} = \frac{\rho_P^{old} V_P^{old}}{\delta t}$$

$$b = S_C V_P + a_P^{old} \Phi_P^{old}$$

$$a_P = a_E + a_W + a_N + a_S + a_H + a_L + a_P^{old} - S_P V_P$$

(2.17 a to i)

The diffusion coefficients D's at the cell faces require interpolation from the adjacent grid nodes, since no values are actually stored at these locations. Usually, an arithmetic mean (*Equation 2.18*) is used to calculate the coefficients; however, for heat transfer problems, a harmonic mean (*Equation 2.19*) is often used instead.

Arithmetic Mean[2]

$$(\Gamma_\Phi)_e = \frac{(\Gamma_\Phi)_E + (\Gamma_\Phi)_P}{2} \tag{2.18}$$

Harmonic Mean[2]

$$(\Gamma_\Phi)_e = \frac{2\,(\Gamma_\Phi)_E\,(\Gamma_\Phi)_P}{(\Gamma_\Phi)_P + (\Gamma_\Phi)_E} \tag{2.19}$$

The convection coefficients C's at the cell faces are calculated by upwinding. Furthermore, there is no need to interpolate velocity values at cell faces since a staggered grid is used.

2.3.4 Discretisation of the Momentum Equation

The integrated and discretised momentum equation is different from the general discretised *Equation 2.16* in two aspects. Firstly, due to the adoption of staggered grid arrangement, the calculation of the diffusion and convection mass fluxes at the cell faces require interpolation. Hence, the coefficients in the integrated momentum equation (or the transport equation) will be different from those of *Equation 2.16.* Secondly, the pressure gradient term in the momentum equation (*Equation 2.3*) has a significant influence on the velocity field of the equation. Hence, it would be inappropriate to *conceal* the pressure gradient into the nominal source term. Thus, this term needs to be added to the integrated transport equation. The integral of the pressure term is expressed as:

$$\iiint_V \nabla p \; dV \tag{2.20}$$

The corresponding finite-difference equations for the momentum equation are:

$$\begin{aligned} a_e u_e &= \sum a_{nb} u_{nb} + b + (p_P - p_E) A_e \\ a_n v_n &= \sum a_{nb} v_{nb} + b + (p_P - p_N) A_n \\ a_h w_h &= \sum a_{nb} w_{nb} + b + (p_P - p_H) A_h \end{aligned} \qquad (2.21 \text{ to } 2.23)$$

[2]Here, it is assumed that the cell face is located midway of the grid nodes P and E.

where *nb* denotes the neighbouring staggered velocities. For example, the velocity u_e has neighbours: w, eE, n and s in two-dimensions. Note eE denotes the velocity at the cell face between nodes E and EE. *b* is defined in the same way as in *Equation 2.16.* The $(p_P - p_E)A_e$ term is the pressure force acting on the *u* control volume, A_e being the area on which the pressure difference acts [Patankar (1980), page 121].

2.3.5 Discretisation of Mass Continuity Equation

As mentioned in *Section 2.2.2.1*, the flow field in the momentum equation (*Equation 2.3*) must satisfy the mass continuity equation (*Equation 2.2*) for the solution to be sensible and converged. Integrating *Equation 2.2* over a finite increment of time and over a finite volume gives:

$$\frac{\partial}{\partial t}\iiint_V \rho \, dV + \iint_S \rho \underline{u} \, dS = 0 \tag{2.24}$$

The discretised mass continuity equation becomes:

$$\begin{aligned}\frac{(\rho_P - \rho_P^{old})V}{\delta t} &+ (\rho uA)_e - (\rho uA)_w + (\rho vA)_n - (\rho vA)_s \\ &+ (\rho wA)_h - (\rho wA)_l = 0\end{aligned} \tag{2.25}$$

2.3.6 Boundary Conditions

The boundary conditions, which specify the physical conditions of the domain at the beginning and during the simulation, can be incorporated into the numerical calculation as linearised source terms (*Equation 2.15*). For PHOENICS, the linearised source is expressed as:

$$S_\Phi = S_C + S_P \Phi_P = \mathrm{C}_\Phi(\mathrm{V}_\Phi - \Phi_\mathrm{P}) \tag{2.26}$$

where C_Φ is the coefficient which is given by:

$$\mathrm{C}_\Phi = - S_P \tag{2.27}$$

and V_Φ is the Value for Φ, and it is given as:

$$\mathrm{V}_\Phi = - S_C / S_P \tag{2.28}$$

2.4 General Numerical Solution Procedure

The finite-difference equations derived in previous sections form a set of simultaneous equations which are solved for, in general, by using a SIMPLE type iterative solution procedure. A brief outline of SIMPLE is presented here.

2.4.1 Pressure and Velocity Coupling

The inter-linking nature or coupling of the momentum equation (*Equation 2.3*) with the mass continuity equation (*Equation 2.2*) has been highlighted in *Sections 2.2.2.1* and *2.3.2.1*. Within this section, the relevance of this coupling to the overall solution procedure is illustrated.

Until early 1970's, the major difficulty encountered by researchers in the development of CFD codes was the calculation of the velocity field when solving for the momentum equation. The trouble lay with the *unknown* pressure field which was hard to determine, but which also dominated the velocity field. The pressure field was difficult to calculate because *it is indirectly specified via the continuity equation* [Patankar (1980)]. Therefore, in order to satisfy the continuity equation, the momentum equation must have the correct pressure field. On the other hand, to approximate the correct pressure field from the continuity equation (expressed in an alternate form), a reasonable converged velocity field must be available. Thus, the pressure and velocity fields are strongly coupled.

In 1972, Patankar and Spalding published an iterative solution procedure called *Semi-Implicit Method for Pressured-Linked Equation* (SIMPLE) which eliminates this problem completely. The SIMPLE procedure [Patankar and Spalding (1972)] and its variants, such as SIMPLER [Patankar (1980)], SIMPLEST [Spalding (1980b)] and SIMPLEC [Van Doormall and Raithby (1984)], have been widely adopted in various CV based CFD codes whether structured or unstructured, for the past decades. The CFD codes, PHOENICS and Harwell-FLOW3D used in the current study, employed SIMPLEST and SIMPLEC respectively.

The implementation of the SIMPLEST algorithm in PHOENICS has been validated extensively on most standard CFD benchmark problems. These include backward facing step, simple laminar and turbulent pipe flow, moving lid cavity, and buoyancy driven flow in rectangular and annular cavities [PHOENICS Beginner's Guide (1987), and PHOENICS Input library]. Markatos and Pericleous (1984) used PHOENICS to obtain solutions of the buoyancy-driven laminar and turbulent flow and heat transfer in a square cavity with differentially heated side walls. The comparisons between their results and the benchmark numerical solutions reported by de Vahl Davis (1983) were in good agreements.

For the unstructured, non-staggered, CV based Harwell-FLOW3D, the SIMPLEC algorithm with the Rhie-Chow interpolation implemented has been validated on a benchmark problem which involves laminar flow in an expanding duct [Burns A D and Wilkes N S (1987)]. Chow (1993) implemented the SIMPLE algorithm with Rhie-Chow interpolation in his unstructured, control-volume based UIFS code to model solidification phenomena in two-dimensions. This code has been validated against a number of benchmark problems which included a buoyancy-driven laminar flow problem and a moving lid cavity problem.

2.4.2 The Pressure and Velocity Corrections

The development of SIMPLE was based upon the staggered grid arrangement which was first proposed by Harlow and Welch (1965) in their MAC method.

The essence of SIMPLE is to correct the pressure field so that the guessed velocity field, which is also subsequently corrected, will get closer to satisfying the continuity equation iteratively. At the beginning of the iterative loop, the momentum equation has only guessed or starred pressure p^* and velocities u^*, v^* and w^*. *Equations 2.21* to *2.23* are re-written as:

$$a_e u_e^* = \sum a_{nb} u_{nb}^* + b + (p_P^* - p_E^*) A_e$$

$$a_n v_n^* = \sum a_{nb} v_{nb}^* + b + (p_P^* - p_N^*) A_n$$

$$a_h w_h^* = \sum a_{nb} w_{nb}^* + b + (p_P^* - p_H^*) A_h$$

(2.29 to 2.31)

To correct the pressure p, it is expressed in the form:

$$p = p^* + p' \tag{2.32}$$

where p' is called the *pressure correction.* Similarly, the corrections for velocity components are expressed as:

$$u = u^* + u' \tag{2.33}$$

$$v = v^* + v' \tag{2.34}$$

$$w = w^* + w' \tag{2.35}$$

Subtract *Equation 2.29* from *2.21* gives:

$$a_e u_e' = \sum a_{nb} u_{nb}' + (p_P' - p_E')A_e \tag{2.36}$$

Similar equations can be obtained for v and w velocity components. The second term $\Sigma a_{nb} u'_{nb}$ of *Equation 2.36* is dropped for reasons which are well documented in [Patankar (1980), page 127]. Then *Equation 2.36* becomes:

$$u_e' = d_e(p_P' - p_E') \tag{2.37}$$

where

$$d_e \equiv A_e / a_e \tag{2.38}$$

Equation 2.37 is known as the *velocity-correction formula* for u, which can also be re-written as:

$$u_e = u_e^* + d_e(p_P' - p_E') \tag{2.39}$$

The velocity correction formulae for other velocity components can be obtained in a similar manner.

$$v_n = v_n^* + d_n(p_P' - p_N')$$
$$w_h = w_h^* + d_h(p_P' - p_H') \qquad \text{(2.40 to 2.41)}$$

So far, the velocity-correction formulae have been derived. For the pressure p, the *pressure-correction formula* is derived from the continuity equation. Substitute all velocity components in *Equation 2.25* with the expressions for velocity from *Equations 2.39* to *2.41*, after rearrangement, a discretised equation for p' is:

$$a_P p'_P = a_E p'_E + a_W p'_W + a_N p'_N + a_S p'_S + a_H p'_H + a_L p'_L + b \tag{2.42}$$

where

$$a_E = \rho_e d_e A_e$$

$$a_W = \rho_w d_w A_w$$

$$a_N = \rho_n d_n A_n$$

$$a_S = \rho_s d_s A_s$$

$$a_H = \rho_h d_h A_h$$

$$a_L = \rho_l d_l A_l$$

$$a_P = a_E + a_W + a_N + a_S + a_H + a_L$$

$$b = \frac{(\rho^{old}_P - \rho_P)V}{\delta t} + [(\rho u^*)_w - (\rho u^*)_e]A_e + [(\rho v^*)_s - (\rho v^*)_n]A_n + [(\rho w^*)_l - (\rho w^*)_h]A_h$$

(2.43 a to h)

The term b represents the 'mass source' or *residuals* for pressure correction. If b is zero, it means that the starred velocities do satisfy the continuity equation and hence p' is also zero.

2.4.3 The SIMPLE Algorithm

The sequence of operations of SIMPLE is listed as follows:

1. Guess the pressure field p^*.
2. Solve the momentum equations (*Equations 2.21* to *2.23*) to obtain u^*, v^* and w^*.
3. Solve the p' equation (*Equation 2.42*).
4. Update p from *Equation 2.32* by adding p' to p^*.
5. Calculate u, v, w, from velocity-correction formulae (*Equations 2.39* to *2.41*).
6. Solve the discretisation equation for other scalar variables Φ if they are coupled to momentum equations. Otherwise, they should be solved after the flow field

has converged.

7. Update the guessed pressure p^* to that of the corrected pressure p and ready for the next iteration by returning to 2. Repeat the whole cycle until either the solution is converged or minimum tolerance reached.

In addition, the rate of convergence of the solution is partially problem-dependent, and due to whether under-relaxation has been applied or not. Under-relaxation is especially important to SIMPLE. This is due to the term $\Sigma_{nb}u'_{nb}$ in *Equation 2.36* which was neglected, and as a consequence the approximation for p' becomes too large, which can result in slow convergence or divergence. So, in order to remedy this problem under-relaxation of the pressure-correction equation of the following form is used:

$$p = p^* + \alpha_p p' \tag{2.44}$$

where α_p is the relaxation parameter, and according to Patankar (1980) the value of 0.8 is usually valid for most problems.

2.4.4 Numerical Solvers

It is worth to notice that SIMPLE is a solution procedure and not a solver. When solving for velocity components or other dependent variables, a numerical solver, such as SOR, is used. Various iterative numerical solvers exist in the literature. The choice of solver is code dependent, for example, PHOENICS has point-by-point Jacobi, slab and whole-field solvers as standard (see [PHOENICS BEGINNER'S GUIDE (1987)] for details), whereas Harwell-FLOW3D has conjugate-gradient and Stone's strongly implicit solver [Stone (1968)] as standard, (see [FLOW3D USER MANUAL (1991)] for details).

2.5 Closure

This Chapter has outlined the general steps required to model the physical phenomena involving fluid flow mathematically using computers. The discretisation of the basic conservation equations and general solution procedures in the control-volume framework, which are subsequently used in later chapters, have also been presented.

Chapter 3

FORMULATION OF THE 3-D SEA FILLING ALGORITHM

3.1 Introduction

The Scalar Equation Algorithm (SEA) which forms the basis for the filling model for simulations of casting applications in three-dimensions, is presented in this Chapter. The mathematical formulation of the algorithm is based upon a procedure described by Liu Jun (1986a, 1986b) and Liu Jun and Spalding (1988), for modelling free surface flow in two-dimensions. The formulation and results for the coupling of filling, heat transfer and solidification are presented in *Chapter 6.*

The procedure proposed by Liu Jun is similar in approach to the Volume-of-Fluid (VOF) algorithm [Hirt and Nichols (1981)], but it models both air and liquid and uses a van Leer scheme [van Leer (1977)] to model and preserve the shape of the liquid metal surface. The basic technique has been validated in two-dimensions for a collapsing liquid column and sloshing in a half-rotated vessel [Liu Jun (1986a, 1986b)].

This Chapter is organised into the following sections:

- *Section 3.2* states the fundamental assumptions which are necessary for the current model to be feasible.
- *Section 3.3* describes the principle of volumetric conservation and the resulting so-called GALA algorithm [Spalding (1974, 1977)].
- *Section 3.4* describes the scalar advection equation which is used to determine whether a cell is occupied by liquid, gas or their mixture.
- *Section 3.5* outlines the van Leer scheme which is used to solve for the scalar advection equation. The scheme can reduce numerical smearing at the gas-liquid interface and convect fluids across the calculation domain.
- *Section 3.6* gives a brief description of the PHOENICS code [CHAM Limited] which provides the platform for the filling model to be developed as an add-on module. The implementation and solution procedure of the filling module attached to PHOENICS are also described.
- *Section 3.7* is the summary of the Chapter.

3.2 Assumptions

In real life, the physical phenomenon under study is most likely to be more complex than the physics which are already known to scientists. So, in order to simulate the phenomenon economically and efficiently, the use of assumptions is necessary to simplify the model. Below is a list of assumptions for the proposed filling model:

1. The fluid flow is Newtonian, laminar and unsteady.
2. The fluid is incompressible.
3. No slip exists between the gas and liquid phases.
4. The fluid properties of both phases remain constant throughout the domain.
5. The free-surface is dominated by convection only.
6. Negligible surface tension exists between the gas and liquid interface.
7. Negligible heat loss exists due to conduction at the gas and liquid interface.
8. The gas temperature remains constant during heat transfer.
9. Negligible heat lost due to radiation to the atmosphere.
10. No internal source or sink of volume.

The significance of these decisions will be discussed in the relevant sections of this Chapter.

3.3 Quasi-Single-Phase Method

In general, the process of mould filling is similar to that of phase change of casting in which molten metal is solidified. Both processes may be classified as Stefan problems [Crank (1984)] in which the boundary between two materials or phases is changing with time. To simulate such processes, the simplest way would be to model them in Eulerian space with a fixed grid (see *Section 1.4, Chapter 1*), rather than using a moving grid in Lagrangian space. However, the use of a fixed grid posed two major immediate problems: tracking the interfaces, as well as the advection of fluid properties across the cell boundaries [Youngs (1982)].

When dealing with two immiscible fluids, such as air and liquid aluminium, in which there is negligible slip between them (Assumption 3), large variations in densities are encountered. The discretised mass continuity equation (*Equation 2.2*) cannot handle such sharp variations in density. An alternative would have been to model the liquid and gas in separate phases and solve them using a two-fluid algorithm like IPSA (Inter-

Phase Slip Analyser [Spalding (1977, 1980a)]) as used by Aldham et al (1982). This would have resulted in solving two sets of Navier-Stokes equations (one for each phase), two sets of continuity equations, a single pressure field, and two sets of enthalpy equations if heat transfer is to be simulated as well. This approach is both uneconomical and unnecessary in terms of computing resources for mould filling since it does not involve any inter-change of mass between air and liquid metal. However, in certain simulations inter-phase slip can occur such as, water evaporating into steam during boiling.

If slip is neglected, the gas-liquid mixtures can be regarded as a single phase fluid [Spalding (1977)]. This leads to the incorporation of a numerical procedure called Gas and Liquid Analyser (GALA) [Spalding (1974, 1977)], into the SEA method. GALA can handle the mixtures of gas and liquid in terms of volume rather than mass. Hence, the two fluids in mould filling can be solved in a quasi-single phase framework. The SEA method is *similar* in approach to the VOF algorithm [Hirt and Nichols (1981)], but *different* in that it accounts for both air and liquid whereas VOF only takes liquid into consideration. Other approaches like the variable blockage technique [Pericleous et al (1984)] and the enthalpy based filling procedure [Swaminathan and Voller (1993)], ignore the air phase. See the Literature reviews in *Section 1.4, Chapter 1* for details.

3.3.1 The Volumetric Continuity Equation

The mass continuity equation (*Equation 2.2, Chapter* 2) can be re-written as:

$$\frac{1}{\rho}\frac{\partial \rho}{\partial t} + \nabla.\underline{u} + \underline{u}\nabla\rho = 0 \tag{3.1}$$

and simplified as:

$$\frac{D(\ln\rho)}{Dt} + \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{3.2}$$

where *D/Dt* is the *substantial derivative* which is defined by the operator $\partial/\partial t + \underline{u}\nabla$, and $\underline{u}$ is the velocity vector. For incompressible fluids in which the densities are not subject to variation in temperature and/or pressure, *Equation 3.2* can be reduced to:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \qquad (3.3)$$

where the substantial derivative in *Equation 3.2* is diminished to zero. *Equation 3.3* implies that the density ρ has no influence on the conservation of volume. Thus, *Equation 3.2* is called the volumetric continuity equation.

3.3.2 The GALA Procedure

The numerical procedure GALA is based upon the volumetric continuity equation (*Equation 3.2*) and is available in PHOENICS as an option. After integrating *Equation 3.2* with respect to δV, the discretised volumetric continuity equation is given as:

$$\frac{V_P - V_P^{old}}{\delta t} + (uA)_e - (uA)_w + (vA)_n - (vA)_s + (wA)_h - (wA)_l = V^* \qquad (3.4)$$

where V^* is the volume source if the substantial derivative is non-zero, for example the volume of liquid metal flowing into the mould. The first term on the left of *Equation 3.4* represents the total rate of volume increase within the cell.

The V_P term becomes zero (ie. the substantial derivative is zero) if the two fluids have constant densities. They can then be regarded as incompressible with little error [Spalding (1977)]. However, this does not mean that the fluid density at any location of the domain is not changing with time. Indeed, the density does vary. This is due to the volume within a cell which always remains the same disregarding whether the cell is occupied by liquid, gas or the mixtures of both.

Once the GALA option is switched on in PHOENICS, the pressure-correction equation (*Equation 2.42, Chapter 2*) is driven by residuals derived directly from the volume flow rates. In other words, the value of density ρ in *Equation 2.42* is unity. The pressure corrections calculated are then applied to the velocity correction formulae (*Equations 2.39* to *2.41*) as described in *Section 2.4.2* of *Chapter 2*. Hence the momentum equation (*Equation 2.3*) will satisfy the volumetric continuity equation (*Equation 3.2*). As a consequence, the properties of the local mixtures of gas and liquid, such as density and viscosity, are needed in the evaluation of relevant terms in the momentum equation. In order to determine these properties, it is essential to be able to track the free surface between the two fluids as accurately as possible. The method used in the current model, is to solve a scalar advection equation which has the dependent variable in terms of volume fraction.

3.4 Advection of a Conserved Scalar Variable

A conserved scalar variable ϕ can be used to distinguish the two fluids by having a value of 1 for liquid, 0 for gas and an intermediate value, such as 0.5, for the gas-liquid interface. *Figure 3.1* depicts a gas-liquid interface, or free surface, formed across a number of cells where ϕ has various intermediate values. This scalar ϕ obeys the volumetric conservation and is expressed in the form:

$$\frac{\partial\phi}{\partial t} + \frac{\partial(\phi u)}{\partial x} + \frac{\partial(\phi v)}{\partial y} + \frac{\partial(\phi w)}{\partial z} = 0 \tag{3.5}$$

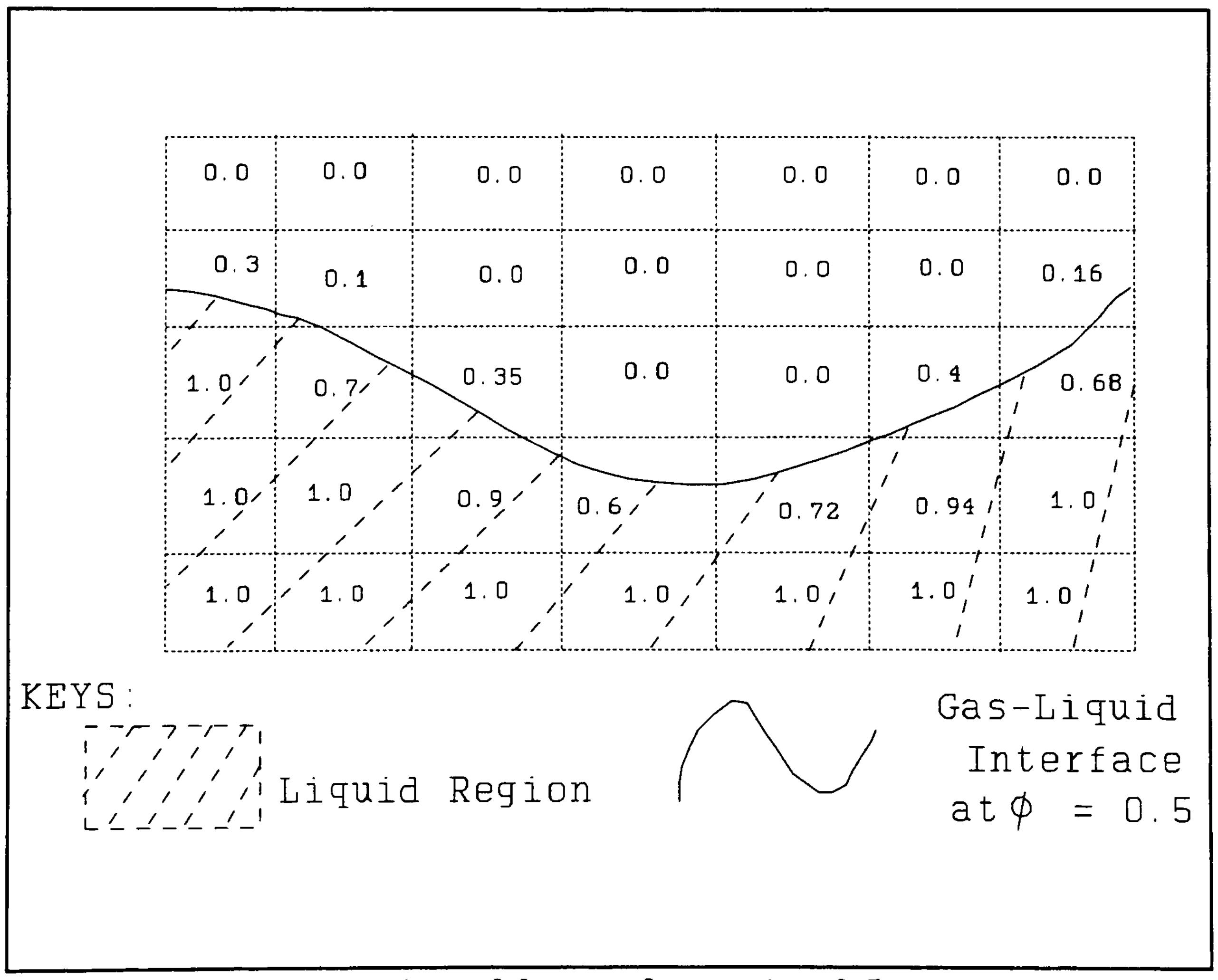


Figure 3.1 Approximation of free surface at ϕ = 0.5

Equation 3.5 is derived from *Equation 2.5* with the source and diffusive terms omitted and ρ set to unity. Since there is no diffusive term in *Equation 3.5*, it implies that ϕ is solely influenced by convection.

3.4.1 Fluid Properties

The fluid density ρ and viscosity μ at the free surface vary with time as the interface position changes relative to the fixed grid. A linear piecewise function is used to determine the instantaneous values of ρ and μ from the values of ϕ by the following relations:

$$\rho_m = \phi\rho_l + (1 - \phi)\rho_a \qquad (3.6)$$

and

$$\mu_m = \phi\mu_l + (1 - \phi)\mu_a \qquad (3.7)$$

where the subscripts ***l***, ***a*** and ***m*** refer to the liquid, air and their mixtures respectively. It is worthwhile to note that, in practice, the 'gas' phase in mould filling is always air. An alternative form of *Equations 3.6* and *3.7*, which uses a special slope-steepening formula to reduce the effects of smearing, is shown as follows:

$$R_m = \max\{R_a, \min[R_l, \lambda(\phi - 0.5)]\} \qquad (3.8)$$

where R is a generic variable which may represent density or viscosity. λ is an arbitrary constant for each property, chosen to be sufficiently large for properties to have their maximum and minimum values in all cells, except those enclosing the interface.

3.4.2 Numerical Diffusion at Gas-Liquid Interface

By imposing limits (in the form of maximum and minium values as shown in *Equation 3.8*) on *Equations 3.6* and *3.7*, the ill effect of numerical diffusion can be reduced. This is required to ensure that fluid properties retain their original values in all but the interface cells. However, interface smearing can still be found to be a problem, especially when first-order differencing schemes, such as upwind, are used to solve *Equation 3.5*. Without a sharp interface, it is virtually impossible to predict the position of the liquid metal surface during mould filling, and this affects the predictions of subsequent heat transfer and solidification. In Hirt and Nichols' VOF algorithm, the well known *donor-acceptor* scheme of Ramshaw and Trapp (1976) is adopted to combat this numerical diffusion at the interface. For the current filling model, the van Leer scheme [van Leer (1977)] is adopted as being the simplest to implement and most economical of the total variance diminishing (TVD) schemes available [Leonard (1988); Bull (1990)].

3.5 Van Leer Scheme for Reducing Numerical Diffusion

The van Leer scheme was first developed as a "shock-capturing" or later known as a TVD scheme for boundary layer model. The method is both second order explicit in time and implicit in space respectively. The monotonicity is preserved by suppressing oscillations due to numerical smearing. An explanation of monotonicity can be found in reference [van Leer (1977), pages 289-292].

3.5.1 Formulation for the Scalar Equation Algorithm

Consider an equation for pure advection of a scalar variable $\phi(\mathrm{x},t)$ by a velocity $\underline{u}(\mathrm{x},t)$ such as *Equation 3.5*. In one-dimension *Equation 3.5* reduces to

$$\frac{\partial \phi}{\partial t} + \frac{\partial(u\phi)}{\partial x} = 0 \tag{3.9}$$

Integrating *Equation 3.9* over a finite time interval δt and cell width $\delta \mathrm{x}$, as shown in *Figure 3.2*, leads to the equation below:

$$\int_{t^n}^{t^{n+1}}\int_{x_w}^{x_e} \frac{\partial \phi}{\partial t}\,dx\,dt + \int_{t^n}^{t^{n+1}}\int_{x_w}^{x_e} \frac{\partial(u\phi)}{\partial x}\,dx\,dt = 0 \tag{3.10}$$

which yields

$$\phi_P^{n+1} = \phi_P^n - \frac{\delta t}{\delta x}(u_e\phi_e - u_w\phi_w) \tag{3.11}$$

where the superscripts $n+1$ and n denote the current and previous time steps respectively; and all other terms are evaluated at the nth time step. Hence *Equation 3.11* is an *explicit* formulation in time. The terms ϕ^{n+1} and ϕ^n are average values of ϕ at the beginning and end of $n+1$th time step, whilst ϕ_e and ϕ_w are average values of ϕ at the east and west cell faces over δt respectively. In addition, the term $u_i\delta t/\delta \mathrm{x}$ in which u_i

can take the value of either u_e or u_w, is the so-called Courant number σ. Its significance will be discussed in *Section 3.5.3*.

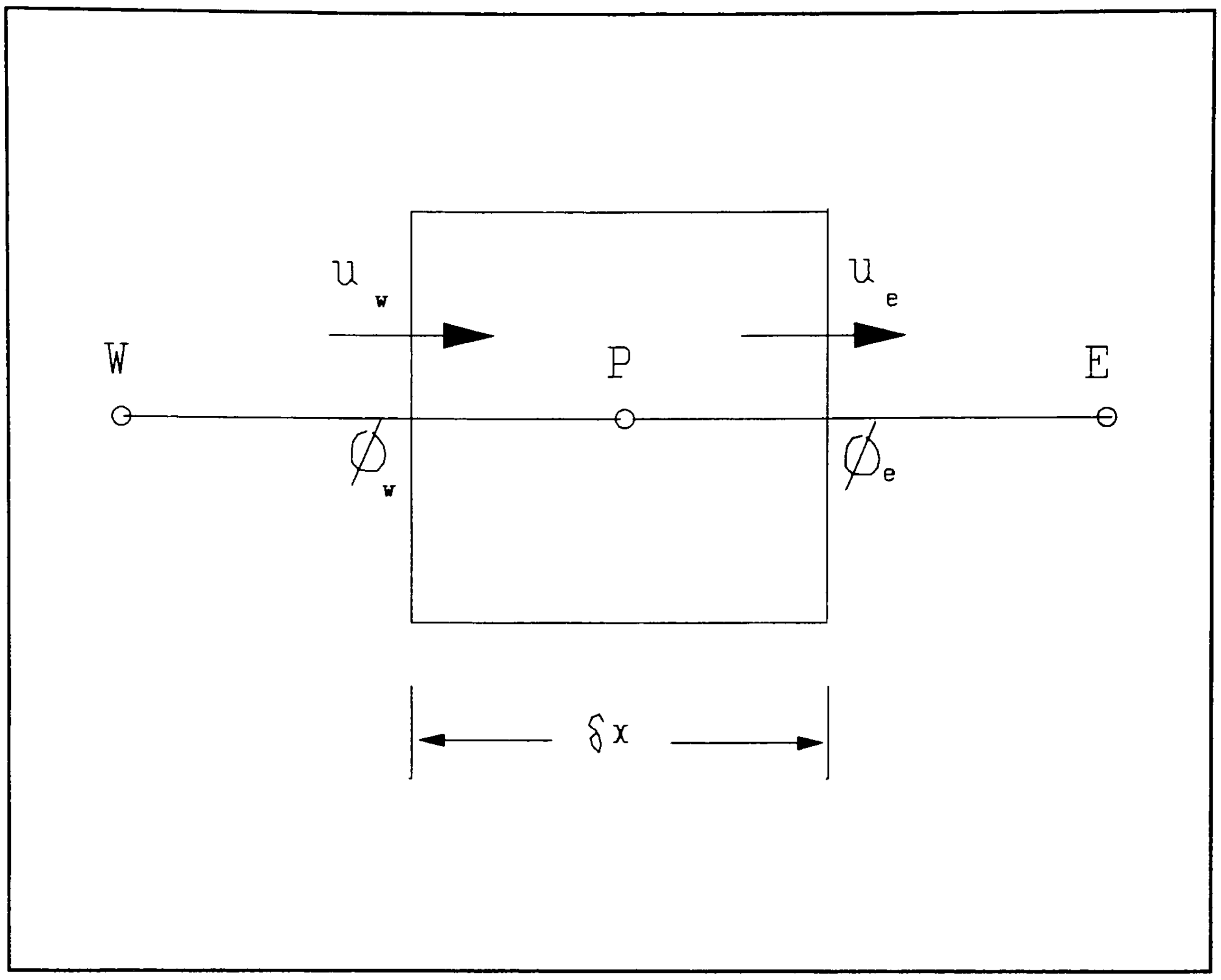


Figure 3.2 A typical 1-D control-volume

The van Leer TVD scheme can be derived in the following steps:

(I) *Provided that an initial-value distribution $\phi(x,t^0)$ is given, determine the cell-averages of ϕ at time δt.* For example, the average of ϕ at the east face of the cell can be defined as:

$$\phi_e = \frac{1}{(u\delta t)} \int_{x_e - u\delta t}^{x_e} \phi(x,t^n)dx \tag{3.12}$$

where $\phi(x,t^n)$ is the profile of ϕ at time $n\delta t$, where n can be $\{0,1,2,....\}$.

(II) *Replace the original initial-value distribution by a piecewise constant function.* The choice of this approximating function will determine the accuracy of the scheme. In this case, the distribution $\phi(x,t^n)$ is approximated by a Taylor series, which is expanded in the direction of the upwind cell P:

$$\phi(x,t^n) = \phi_P + \left[\frac{\partial\phi}{\partial x}\right]_P (x - x_p) + \textit{Higher Order Terms} \tag{3.13}$$

where the gradient $\partial\phi/\partial x$ is evaluated at the centre of cell P.

(III) *With the approximated initial-values (Equation 3.13, ignoring all higher order terms), integrate Equation 3.13 over a finite time step δt.* This is done by shifting the distribution $\phi(x,t^n)$ over a distance $u\delta t = \sigma\delta x$ along the x-axis. That is to say,

$$\phi(x,t^{n+1}) = \phi(x - \sigma\delta x, t^n) \tag{3.14}$$

Equation 3.14 states that the shape of ϕ does not change after travelling a distance of $\sigma\delta x$. As a consequence, a second order representation of ϕ_e can be derived from *Equation 3.12*, by substituting *Equation 3.13* into *3.12* and ignoring the higher order terms.

For $u_e > 0$,

$$\phi_e = \phi_P + \left[\frac{\partial\phi}{\partial x}\right]_P \frac{\delta x}{2}\left(1 - \frac{u\delta t}{\delta x}\right) \tag{3.15}$$

and for $u_e < 0$,

$$\phi_e = \phi_E - \left[\frac{\partial\phi}{\partial x}\right]_E \frac{\delta x}{2}\left(1 + \frac{u\delta t}{\delta x}\right) \tag{3.16}$$

Similar expressions can also be derived for ϕ at cell-faces in both y and z directions respectively.

(IV) *Repeat Steps (I) to (III) for the next time step.*

The gradient $\partial\phi/\partial x$ was approximated by central difference discretisation as shown below:

$$\left[\frac{\partial\phi}{\partial x}\right]_P = \frac{(\phi_E - \phi_W)}{2\delta x} \tag{3.17}$$

In order to preserve the monotonicity of a sequence of cell averages, van Leer (1977) suggested that *the linear function (Equation 3.13) must not take values outside the range spanned by the neighbouring cell averages.* A successful monotonicity condition for the gradient of ϕ reported by van Leer (1977) is shown as follows:

If $sgn(\delta_e) = sgn(\delta_w) = sgn(\ 1/2(\delta_e + \delta_w))$,

$$\left[\frac{\partial\phi}{\partial x}\right]_P = \frac{2}{\delta x}\, sgn(\delta_e)\ \min\left[\,|\delta_e|,\frac{1}{2}(\,|\delta_e|+|\delta_w|),\ |\delta_w|\,\right] \tag{3.18}$$

where $\delta_e = \phi_E - \phi_P$, $\delta_w = \phi_P - \phi_W$; *sgn* means 'the sign of', and $sgn(\delta_e) = +1$ if $\delta_e \geq 0$, otherwise $sgn(\delta_e) = -1$.

Should an extremum occur, that is when $\delta_e.\delta_w < 0$ the gradient of ϕ becomes zero as below:

$$\left[\frac{\partial\phi}{\partial x}\right]_P = 0 \tag{3.19}$$

This monotonicity condition ensures that if an initial-value distribution is monotonic, after it has been advected numerically over a distance $\sigma\delta x$, it remains monotonic as before.

3.5.2 Application to 3-D Filling Model

So far, the formulation of the one-dimensional SEA method has been addressed. To solve a three-dimensional scalar advection equation (*Equation 3.5*), the one-dimensional formulation is repeated for the cell-faces in x, y and z directions respectively; and according to Young (1982) and Liu Jun (1986b), the solution has to be performed in separate steps. The convection of ϕ is split into three parts: x, y and z. In order to avoid bias, the sequence of advection is alternated according to the number of time-step using the MOD function in FORTRAN programming language, as follows:

```
IF( MOD( ISTEP-1,3) .EQ. 0 ) THEN
        Advect X-direction first, Y-direction second, Z-direction third
IF( MOD( ISTEP-1,3) .EQ. 1 ) THEN
        Advect Y-direction first, Z-direction second, X-direction third
IF( MOD( ISTEP-1,3) .EQ. 2 ) THEN
        Advect Z-direction first, X-direction second, Y-direction third
```

The value of new ϕ_p after each advection sequence, can be evaluated using the principle of volume conservation. Suppose, the advection sequence starts with the x-direction first:

Perform the X-direction convection,

$$\phi_P^* = \frac{V_P^n \phi_P^n + \delta V_w \phi_w - \delta V_e \phi_e}{V_P^*} \tag{3.20}$$

then perform the Y-direction convection,

$$\phi_P^{**} = \frac{V_P^* \phi_P^* + \delta V_s \phi_s - \delta V_n \phi_n}{V_P^{**}} \tag{3.21}$$

followed by the Z-direction convection,

$$\phi_P^{n+1} = \frac{V_P^{**} \phi_P^{**} + \delta V_l \phi_l - \delta V_h \phi_h}{V_P^{n+1}} \tag{3.22}$$

where * and ** denote intermediate values and

$$\begin{aligned}
\delta V_e &= u_e \delta t \delta y \delta z \\
\delta V_w &= u_w \delta t \delta y \delta z \\
\delta V_n &= v_n \delta t \delta x \delta z \\
\delta V_s &= v_s \delta t \delta x \delta z \\
\delta V_h &= w_h \delta t \delta x \delta y \\
\delta V_l &= w_l \delta t \delta x \delta y
\end{aligned} \qquad \text{(3.23 to 3.28)}$$

To ensure that the volumetric continuity equation (*Equation 3.2*) is adhered to, the following check can be performed. Let

$$\begin{aligned}
V_P^* &= V_P^n + \delta V_w - \delta V_e \\
V_p^{**} &= V_P^* + \delta V_s - \delta V_n \\
V_P^{n+1} &= V_P^{**} + \delta V_l - \delta V_h
\end{aligned} \qquad \text{(3.29 to 3.31)}$$

then

$$V_P^{n+1} = V_P^n + (\delta V_w - \delta V_e + \delta V_s - \delta V_n + \delta V_l - \delta V_h) = V_P^n \tag{3.32}$$

When the volumetric continuity equation (*Equation 3.2*) is satisfied the volume-fluxes inside the brackets ( ) converge to zero.

3.5.3 Courant Criterion

Since the van Leer TVD scheme is explicit in time, the time-step used in the simulations is subjected to the so-called Courant criterion for the sake of numerical stability. For practical problems, the Courant number σ is limited to $|\sigma| \leq 1$.

In the case of three-dimensional mould filling, the time-step δt is subject to the following restriction:

$$\delta t < \min\left\{ \left|\frac{\delta x}{u}\right| , \left|\frac{\delta y}{v}\right| , \left|\frac{\delta z}{w}\right| \right\} \tag{3.33}$$

which implies that the gas-liquid interface must not travel more than a full cell within a time-step, δt.

This condition may cause extremely small time-steps to be used in some situations. For example in the simulation of the rapid filling of die-castings, time steps right down to 10^{-5} second were required.

3.6 Implementation and Solution Procedure

For the 2-D implementation of the SEA algorithm, several benchmark problems used by Liu Jun (1986b) have been performed to validate the algorithm. The results for one of the benchmark problem can be found in Appendix A. (However, it is not the objective of this study to rework all the cases reported by Liu Jun.)

The 3-D mould filling model - 'FILL' has been implemented successfully as a 'stand-alone' modular attachment to PHOENICS [CHAM Limited] solely via its user routines (ie. GROUND subroutine). The decision to employ PHOENICS for this study was based upon a number of factors. The major ones were availability, prior knowledge, and maturity and reliability of the code which has been released to industrial and academic clients since 1981.

3.6.1 Overview of PHOENICS

PHOENICS which stands for Parabolic, Hyperbolic Or Elliptic Numerical Integration Code Series, is a general purpose three-dimensional fluid-flow and thermal code. A wide range of flow scenarios can be simulated by PHOENICS, whether they are steady-state or transient, laminar or turbulent, single- or two-phase, with heat-transfer or not; within a framework of either Cartesian, polar or body-fitted coordinates (BFC).

All these phenomena are described in the form of algebraic finite-domain equations (as those outlined in *Chapter* 2) in terms of enthalpy (or temperature), velocity, pressure and other physical quantities. These algebraic equations are then solved for iteratively using numerical solvers: such as Jacobi point-by-point, slab-by-slab, or whole field [PHOENICS Beginner's Guide (1987)] under the framework of the SIMPLEST solution algorithm [Spalding (1980b)] for one-phase flows, or under the framework of IPSA [Spalding (1977, 1980a)] for two-phase flows. Note that SIMPLEST is a variant of SIMPLE [Patankar and Spalding (1972)].

The resulting iterative solution procedure is complicated, involving a multi-stage sequence of adjustment of values before a satisfactory solution can be obtained. Within an iterative solution cycle, the most noteworthy concepts are *slabwise*, *sweeps* and *whole-field solution.*

A *slab* consists arrays of cells (in x- and y-directions) which share the same z coordinates. Hence, a three-dimensional computational domain in cartesian coordinates, can be considered being made up of a set of slabs; just like a loaf of sliced bread. Many mathematical operations and cycles of adjustments can be performed over a single slab before attention is moved to the next slab in a higher z coordinate. These adjustment cycles are referred to as slabwise solutions; see the *Figure 3.3* for a structure chart which shows how a slabwise solution cycle is used to solve a set of elliptic

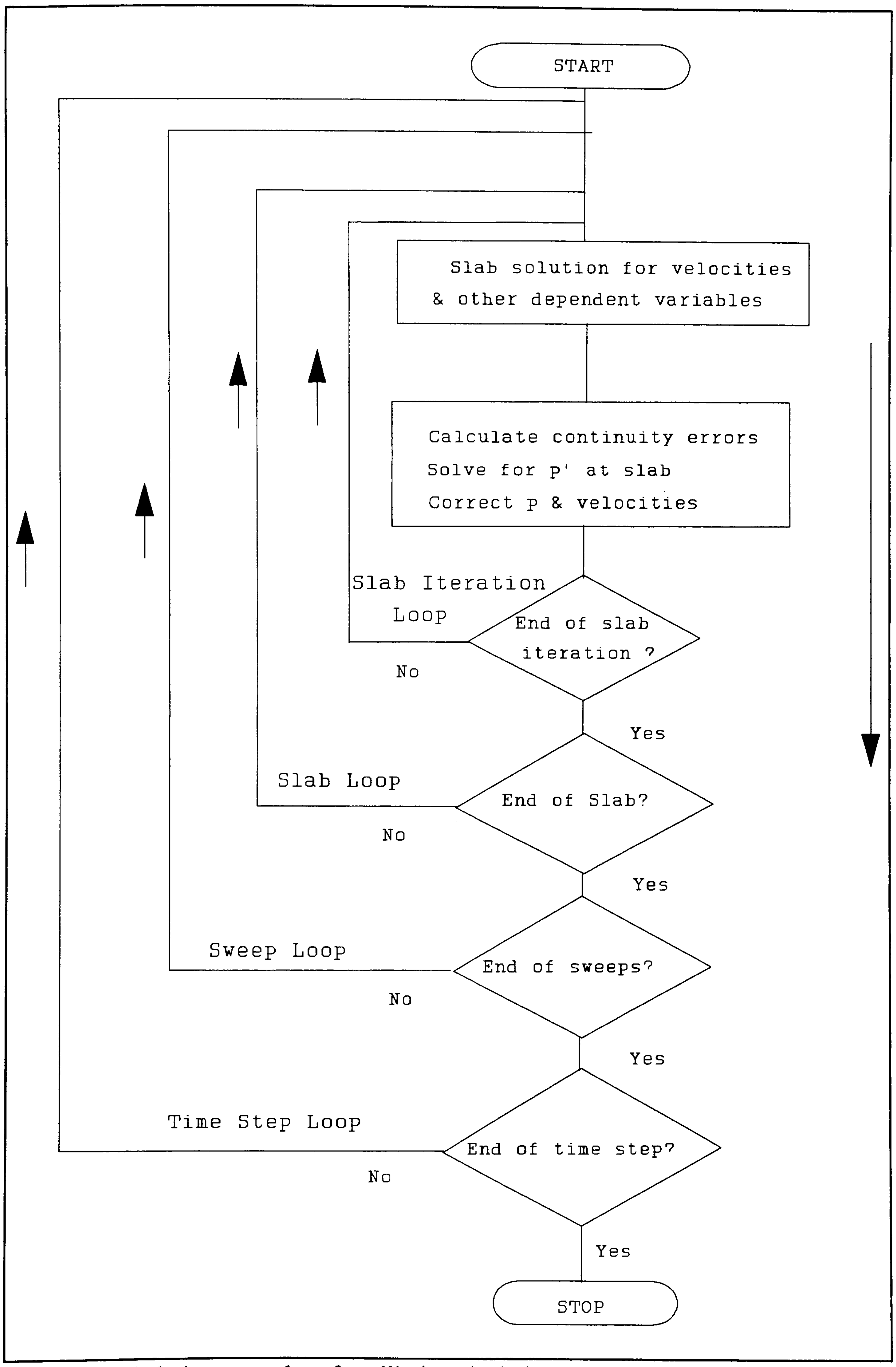


Figure 3.3 Solution procedure for elliptic calculations with pressure solved slabwise.

equations.

A *sweep* is a set of slabwise operations conducted in sequence from the lowest to the highest slab in z-direction.

Whole-field is a procedure which differs to slabwise solution by taking the z-direction links into account simultaneously with x- and y-direction links. See the *Figure 3.4* for a structure chart of solution scheme using whole-field. In case when z-direction links dominate, such as in fire simulations [Hoffmann (1990)], the whole-field solution is more effective, particularly for pressure calculation.

When comparing the slabwise and whole-field solutions, the former requires more sweeps to converge than the latter, but the latter demands more computer memory than the former. In addition, special settings of relaxation parameters may be needed to ensure convergence. This is due to the nature of the equations being solved, which are, in general, non-linear and highly coupled. Therefore, relaxation (usually *under-relaxation*) can be viewed as a useful device to help the solution to converge when iterative procedures are applied. However, the converged solution obtained should not be viewed as a product of the relaxation settings being used. Relaxation merely *helps* to accelerate the solution to reach convergence if over-relaxation is specified, or to prevent the solution to diverge by slowing down changes from one iteration to next if under-relaxation is chosen. There are two under-relaxation options available in PHOENICS, namely the *false time step* and *linear under-relaxation* methods.

3.6.2 Incorporation of FILL into PHOENICS

The necessary FORTRAN coding needed to simulate mould filling has been programmed in a stand-alone module 'FILL' attached to PHOENICS versions 1.4 and 1.6 as shown in *Figure 3.5*.

The interface between the PHOENICS' Earth module (ie. the main numerical solvers) and FILL is done via the subroutine 'GROUND'.

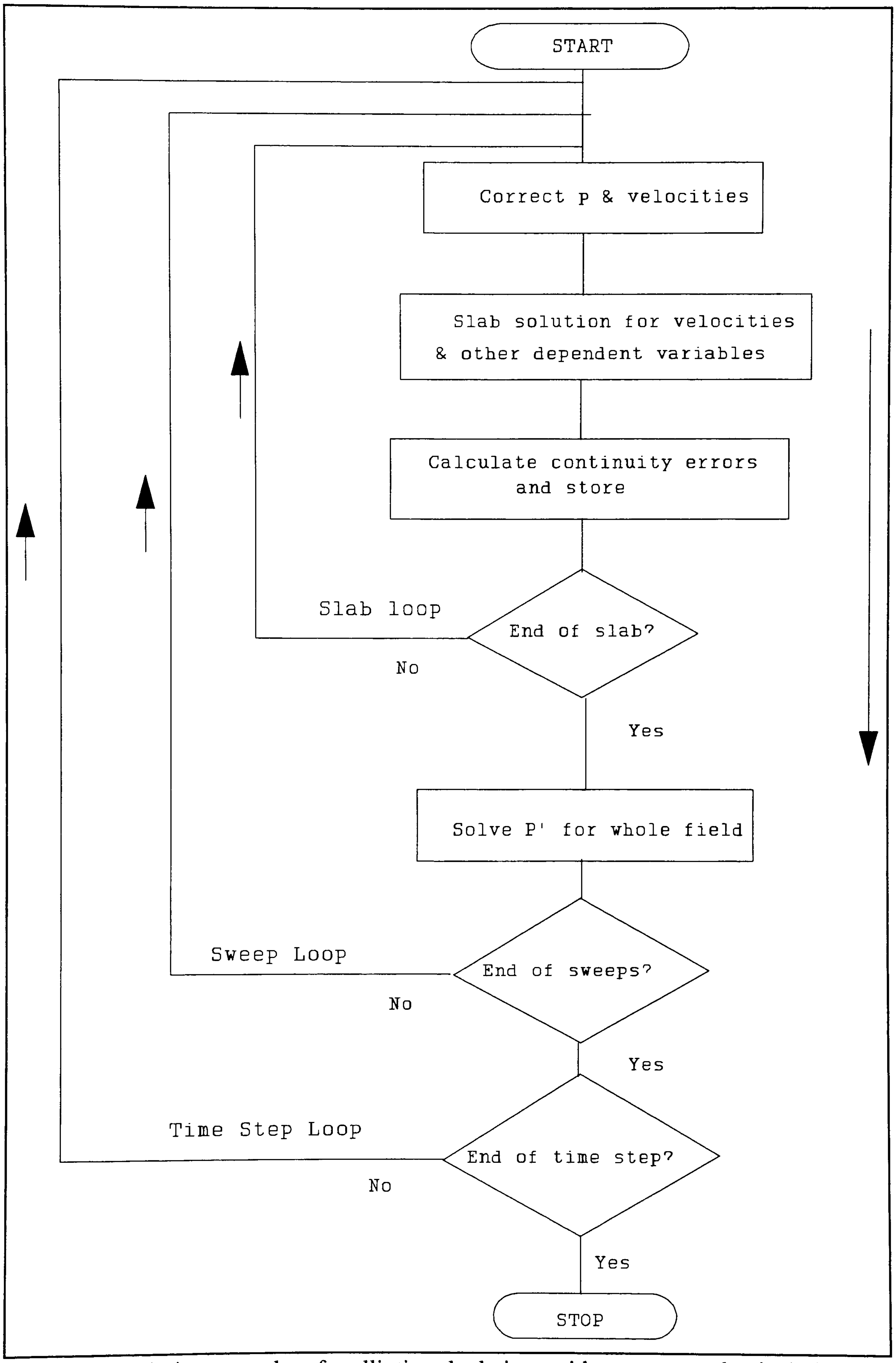


Figure 3.4 Solution procedure for elliptic calculations with pressures solved whole-field.

The geometry of the problem to be simulated, time-step size, its initial and boundary conditions, physical properties and other run-time conditions are specified in the so-called 'Q1' file.

The data stored in Q1 files are subsequently read and validated by another PHOENICS' module called 'SATELLITE' (which is an interactive data pre-processor). The inlet and outlet boundary conditions are also required to be specified in the GROUND and FILL modules respectively.

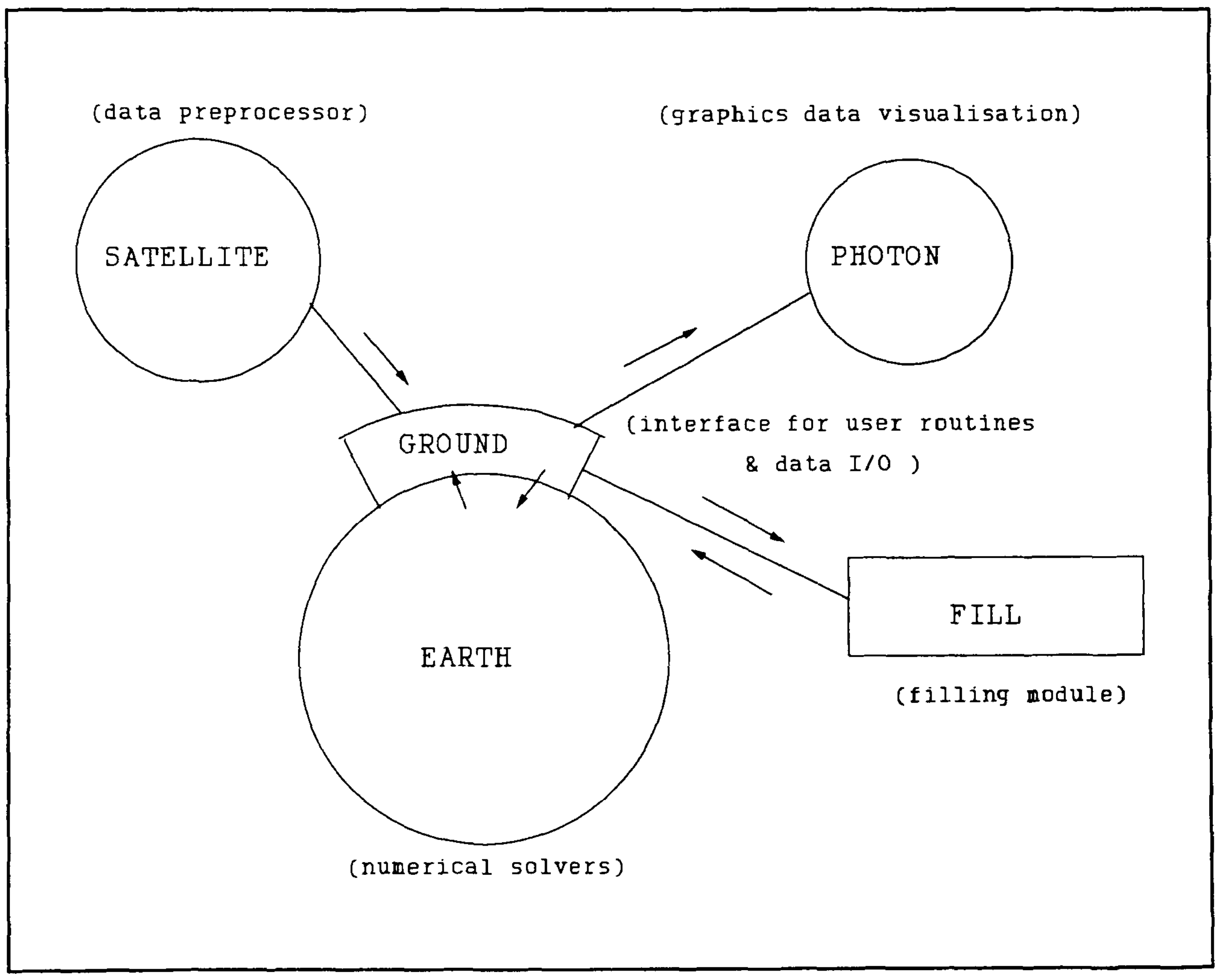


Figure 3.5 A brief outline of PHOENICS program structure

The results of the simulation can be viewed graphically via the PHOENICS' PHOTON interactive graphics post-processor. However, a number of more sophisticated graphical visualisation packages, such as AVS, Wavefront and Femview which are available at the University of Greenwich, can also be used to visualise phenomena modelled by PHOENICS.

3.6.2.1 Solution Procedure for FILL

The flow chart (shown in *Figure 3.6*) illustrates the solution procedure of FILL. The computational cycles within Earth start by:

1. calculating the amount of volume flux entered into the computational domain as inflow via Group 8 Section 7 -- Gala source.

2. Then control is moved to Group 9 Sections 1 and 6 to determine the new mean density and viscosity distributions respectively, ie. evaluating *Equations 3.6* and *3.7*.

3. Initialise ϕ^{old} to ϕ^{new} of previous time-step for every cell at the beginning of the *Sweep-loop* in Group 19 Section 3.

4. Solve the flow field for u,v,w and p with initial and boundary conditions specified via Q1 under the framework of SIMPLEST. The velocities are solved by a slabwise method while the pressure is solved whole-field.

5. In Group 19 Section 6 (end of iz slab) velocities (u, v and w) of each slab are copied into local arrays to be used by FILL later. Set the outlet velocities at cell boundaries since PHOENICS does not specify outlet velocity at boundaries. The outlet velocity used for FILL are derived from the neighbouring cell normals to the outlets.

6. In Group 19 Section 7 (finish of a sweep) set inlet velocity at cell boundaries (to be used by FILL) since PHOENICS does not store velocities at boundaries.

7. Call FILL to calculate ϕ^{new*}; the latest ϕ at end of current sweep using the method described in this Chapter.

8. Update ϕ^{new} to ϕ^{new*}. Repeat steps 1 to 8 until the number of sweeps specified has been reached.

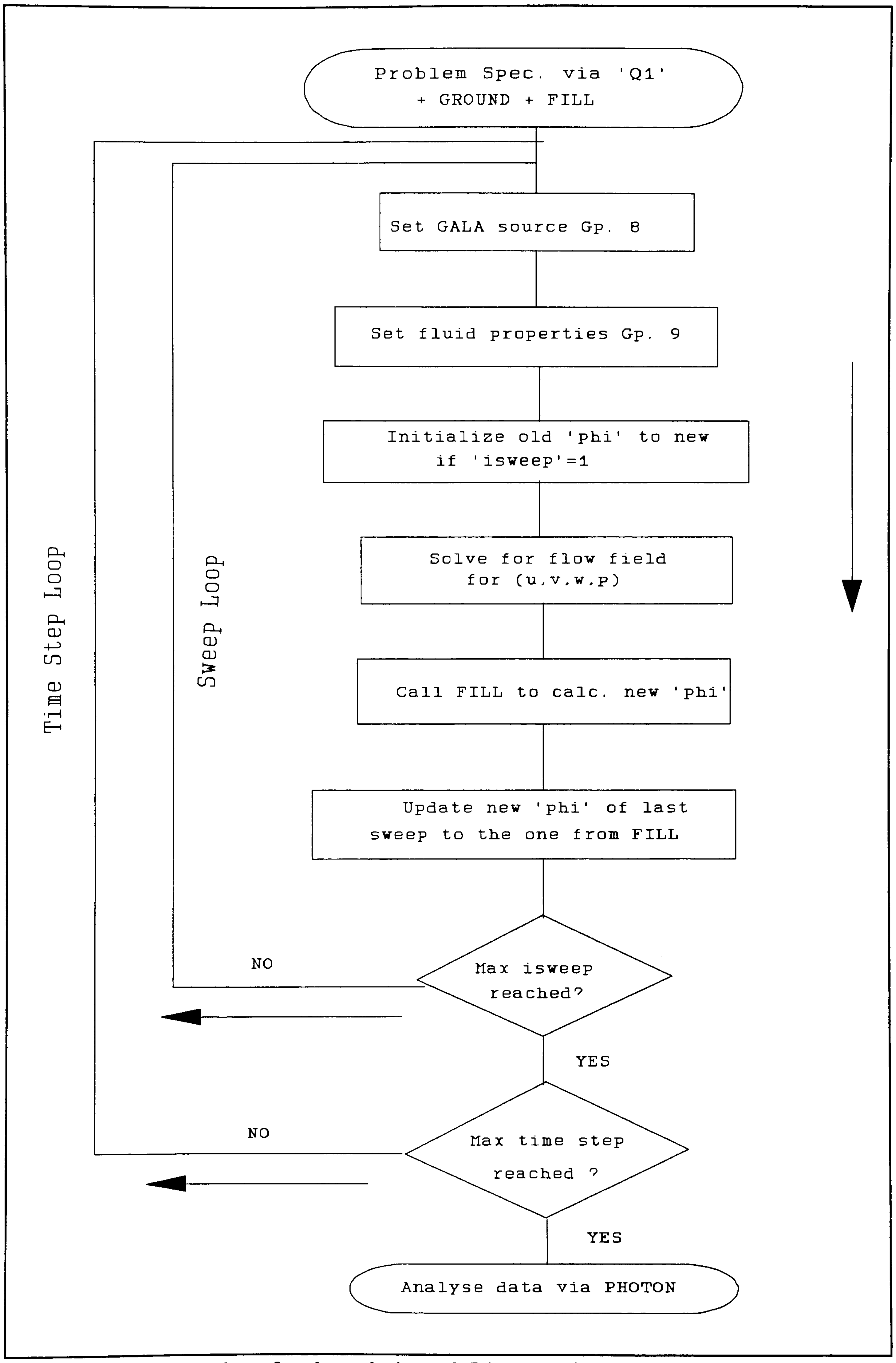


Figure 3.6 A flow chart for the solution of FILL attaching to PHOENICS

9. In Group 19 Section 8 (finish of time step) if the number of maximum time-steps has not been reached, repeat steps 1 to 8.

This is the basic solution procedure for FILL without heat-transfer and solidification.

3.7 Closure

In this Chapter, the use of a scalar variable, ϕ, together with the van Leer scheme to track the gas-liquid interface has been explained. The application of the Scalar Equation Algorithm to model filling processes in three-dimensions using PHOENICS as a numerical solution platform has also been outlined. The results of simulations using this model can be found in *Chapters 4 to 6.*

Chapter 4
FREE SURFACE FILLING - VALIDATION AND RESULTS

4.1 Introduction

In the previous two chapters, the mathematical basis and formulation for the free surface filling model based upon the SEA algorithm has been discussed in detail. Results of simulations for industrial and academic applications are presented in this Chapter.

This Chapter is organised into the following subsections:

- *Sections 4.2 and 4.3* describe two validation cases. The first case involves comparisons of results for simulation of water being filled into a rectangular mould cavity under high pressure against experimental results. The second case is focusing on simulations of mould filling of aluminium under gravity.
- *Section 4.4* describes two simulations of an industrial sand casting.
- *Section 4.5* describes simulation of another industrial example - a step wedge in both two and three dimensions.
- *Section 4.6* describes simulation of an academic or test case used by other researchers. A qualitative comparison of the results obtained by the SEA method with those calculated by SOLA-VOF method [Lin and Hwang (1988)], and the MAC method [Hwang and Stoehr (1987)] are presented.
- *Section 4.7* is the closure of this Chapter.

4.2 Validation case I: Pressure Die Casting - water experiments

In 1983, Booth and Allsop presented a paper entitled: *Cavity Flow Studies at BNF Metals Technology Centre Using Water Modelling Techniques* to the audience of the 12th International Die Casting Congress and Exposition, Minneapolis, USA. Their paper was aimed at investigating the effects of flow patterns caused by using different designs of runner systems on the quality of the finished die casting products.

Their experiments were carried out using a Perspex die, a fast precision camera and water. Hot liquid metal was not used due to reasons of safety, costs of experimental equipment and limitation of technology available at that time. Nowadays, sophisticated computer-controlled machines capable of displaying three-dimensional X-ray images of hot molten metal filling a mould cavity in real time are available from Japan. These however, come at a cost upwards in the region of £300K. One such machines was acquired by the Interdisciplinary Research Centre (IRC) at the University of Birmingham in 1993 and experimental results are anticipated in the near future.

4.2.1 Problem specification

Figure 4.1 shows the arrangement of the Perspex experimental kit used by Booth and Allsop (1983). Cavities were machined on the inside faces of two pieces of Perspex of dimensions 50 mm x 190 mm x 145 mm. The Perspex was then held together rigidly by aluminium clamps. The fluid, in this case water dyed with ink, was then rammed into the cavity by a plunger through the runner. A detailed diagram of the Perspex die showing the flat plate cavity with parallel runner, overflow and vents can be found in *Figure 4.2*.

The size of the cavity for simulation was 150 mm long, 100 mm wide and 1.5 mm thick. The overflow had the same volume as the cavity. The gate, through which the water was pumped in, was 0.5 mm thick and it ran the full length of the cavity giving an inlet area equal to the cross-sectional area of the runner (which is 70.0569 mm^2).

4.2.2 Computational Details

A three dimensional structured mesh of size 6 x 19 x 17 = 1938 cells was used with the SEA filling algorithm to simulate the problem described above. The geometry and the mesh of the control volume domain can be seen in *Figure 4.3*. The fluid was set to come in with an initial velocity of 30 m/s from the inlet at the bottom of the runner. There were three outlets or vents situated at the other end of the cavity, just above the overflow tank. These allowed the air which was originally present to be expelled. Summaries of the material properties and initial and boundary conditions can be found in *Tables 4.1* and *4.2* respectively.

The fluid was forced into the cavity via the gate along the runner. The air originally within the cavity was expelled by the fluid, and exited via the three vents (or outlets) at the other end of the mould cavity, just above the overflow.

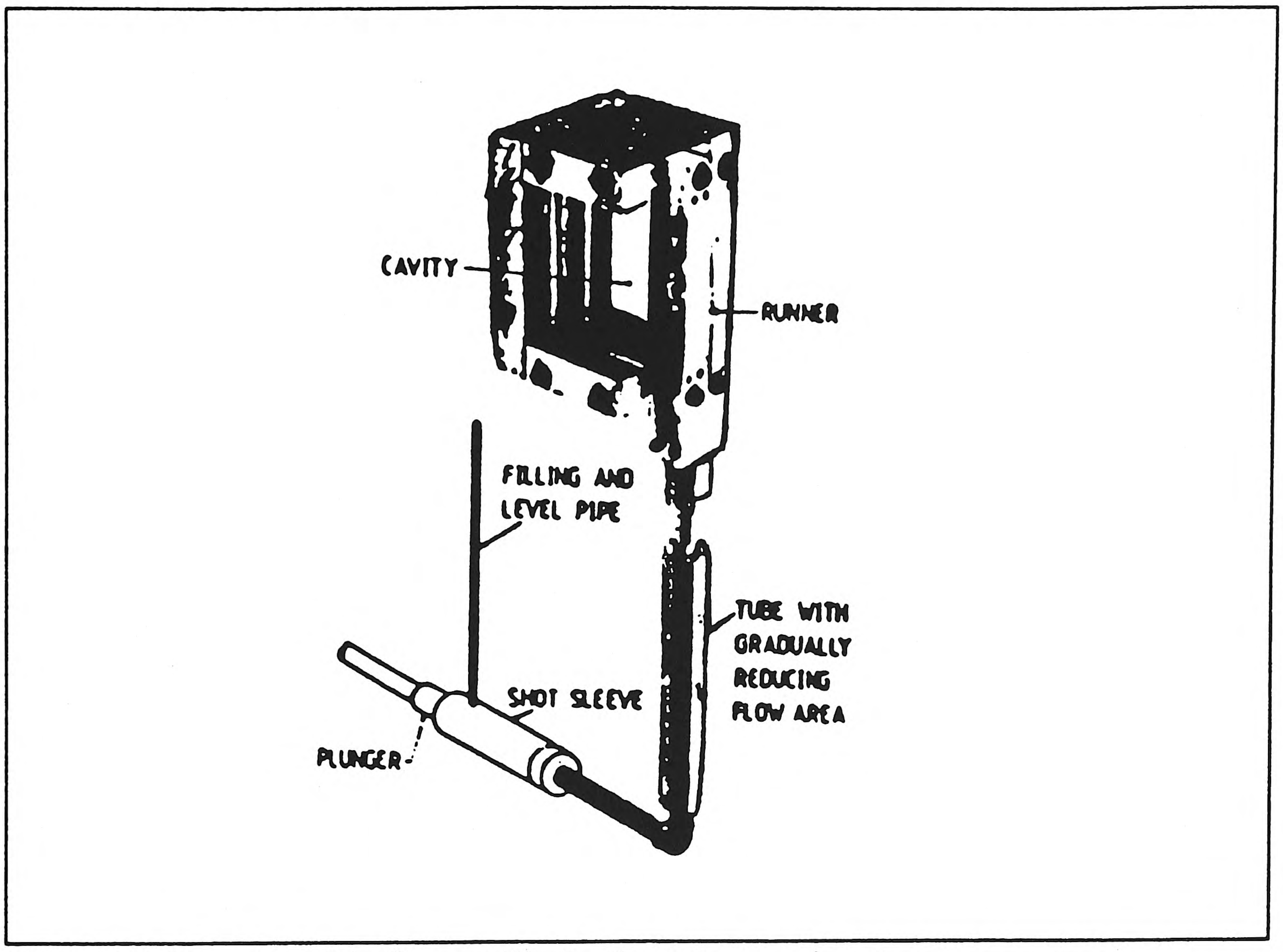


Figure 4.1 The setup of the Perspex Die for the water experiments.

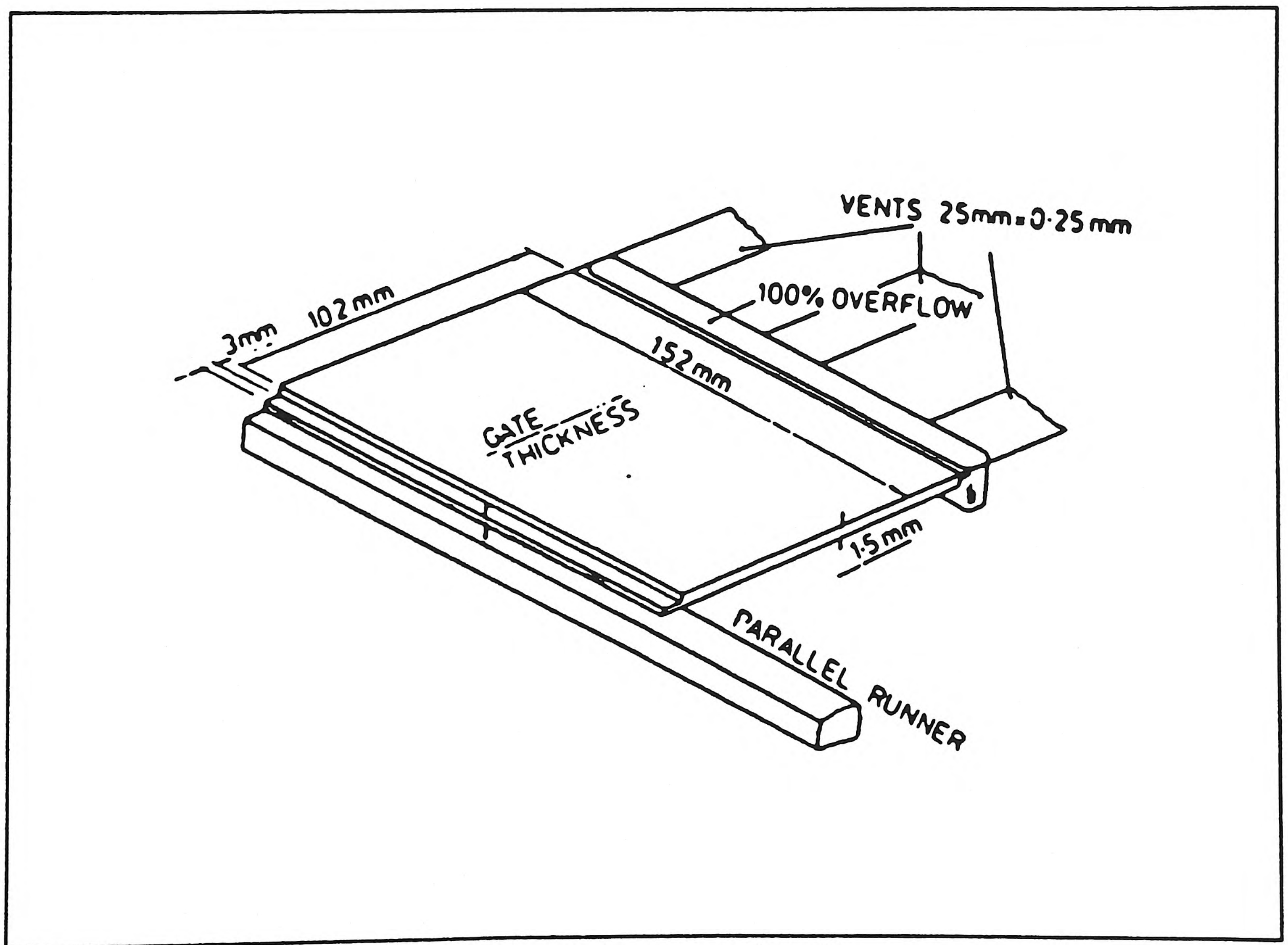


Figure 4.2 Schematic diagram of the vertical runner and cavity.

	Water	Air
Density, ρ (Kg/m^3)	1000.0	1.0
Kinematic Viscosity, ν (m^2/s)	10^{-6}	10^{-5}

Table 4.1 Material properties for validation case I.

The run-time parameters applied in this simulation are described as follows. A total of 3300 time-steps of δt= 0.000001 second were used to simulate the filling of the die cavity of 0.033 second in real time. To ensure convergence of the solution 60 sweeps were used. To aid convergence, a false time-step relaxation factor of 0.003 was used when solving the momentum equations for u,v and w. No relaxation was applied to the evaluation of the liquid volume fraction ϕ, since the value of ϕ was indirectly determined from the values of cell velocities. *Table 4.3* summaries the run time conditions used for convergence with the filling model attached to PHOENICS (version 1.4) when running on a Norsk Data ND5900 mini-computer. A total run-time of about 121 hours was required to complete the simulation. For CFD codes, the ND5900 performs at 0.5 Mflop.

4.2.3 Results

At the beginning of the simulation, water flows into the runner against gravity at the lower left corner of the domain (see *Figure 4.3*) and is later injected into the die cavity via the gate along the runner. Because the simulation was performed in three-dimensions, the *slope* formed by the isosurface representing the air and water interface at ϕ = 0.5 can clearly be seen in *Figures 4.4* to *4.7*.

Figure 4.4 depicts the photographic image of the water experiment and the profile of the filling pattern obtained by the model after 0.006 second. The simulated water pattern shown on the right hand side, is represented by an isosurface plot with the

Initial conditions:

Inflow velocity: 30 m/s

Outlets (vents): Pressure at outlet boundary is prescribed to zero.

Boundary conditions:

Walls: impermeable, constant temperature, and non-slip

or zero velocity at walls is assumed.

Gravity acting downward: 9.81 ms^{-2}

Other information:

Liquid volume flow rate : $2.1x10^{-3}$ m^3/s

Volume of computational domain: $5.716x10^{-5}$ m^3

Expected fill-up time : 0.027 s

Table 4.2 Initial and boundary conditions for validation case I.

Simulation of real time: 0.033 s

Time-step size: 10^{-6} s

Total number of time-steps: 3300

Number of sweep per time-step: 60

Residual tolerance for velocities and pressure : 10^{-5}

Relaxation factors:

Linear: pressure (0.8)

False time-step: u (0.003), v (0.003), w (0.003)

Table 4.3 Run-time parameters for validation case I.

velocity vector plot showing the movement of liquid and air within the cavity superimposed. Both the experimental and simulated results clearly show how water enters into the die cavity through the gate and along the parallel runner. The filling pattern simulated by the model matches qualitatively with that obtained by experiment, in terms of trend and profile.

In *Figure 4.5* more fluid moves into the cavity, especially at the top after just 0.009 second. Air is being expelled via the air vents as shown by the velocity vectors. The profiles of both experimental and simulated results are in close agreement, except at the flow front. In the experiment, the front has not reached the overflow at the top right corner of the cavity whereas in the simulated result it has. This could be due to the fact that turbulence was not accurately modelled or that slight numerical smearing may exist at the interface.

After 0.012 second, about half of the die cavity is filled with water as shown in both experimental and simulated results in *Figure 4.6*. The flow angle between the gate and the direction of the flow was increased which can be seen by observing the changes in velocity vectors from previous figures. Water is forced out of the cavity and the overflow at the top right corner of the cavity via a vent. An obvious difference between the experiment and the simulations is that the amount of water which flowed into the overflow is not the same.

After 0.03 second, the cavity should have been filled completely according to a rough estimate shown in *Table 4.2*; that is, the expected fill up time is calculated by dividing the total volume of the domain by the mass flow rate at the inlet. However, this is not the case as depicted by both experimental and simulated results in *Figure 4.7*. This could be due to the amount of water which has prematurely exited the cavity through the vents, and the possible effect of air which is being compressed at the lower part of the cavity.

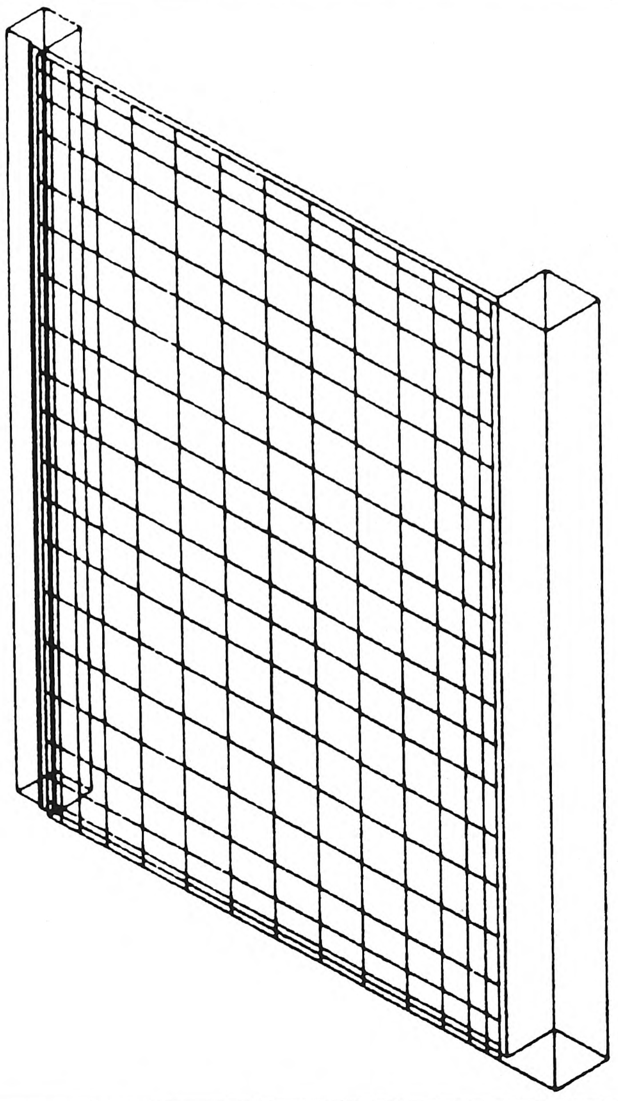


Figure 4.3 The 3-D mesh used for the pressure die casting simulation.

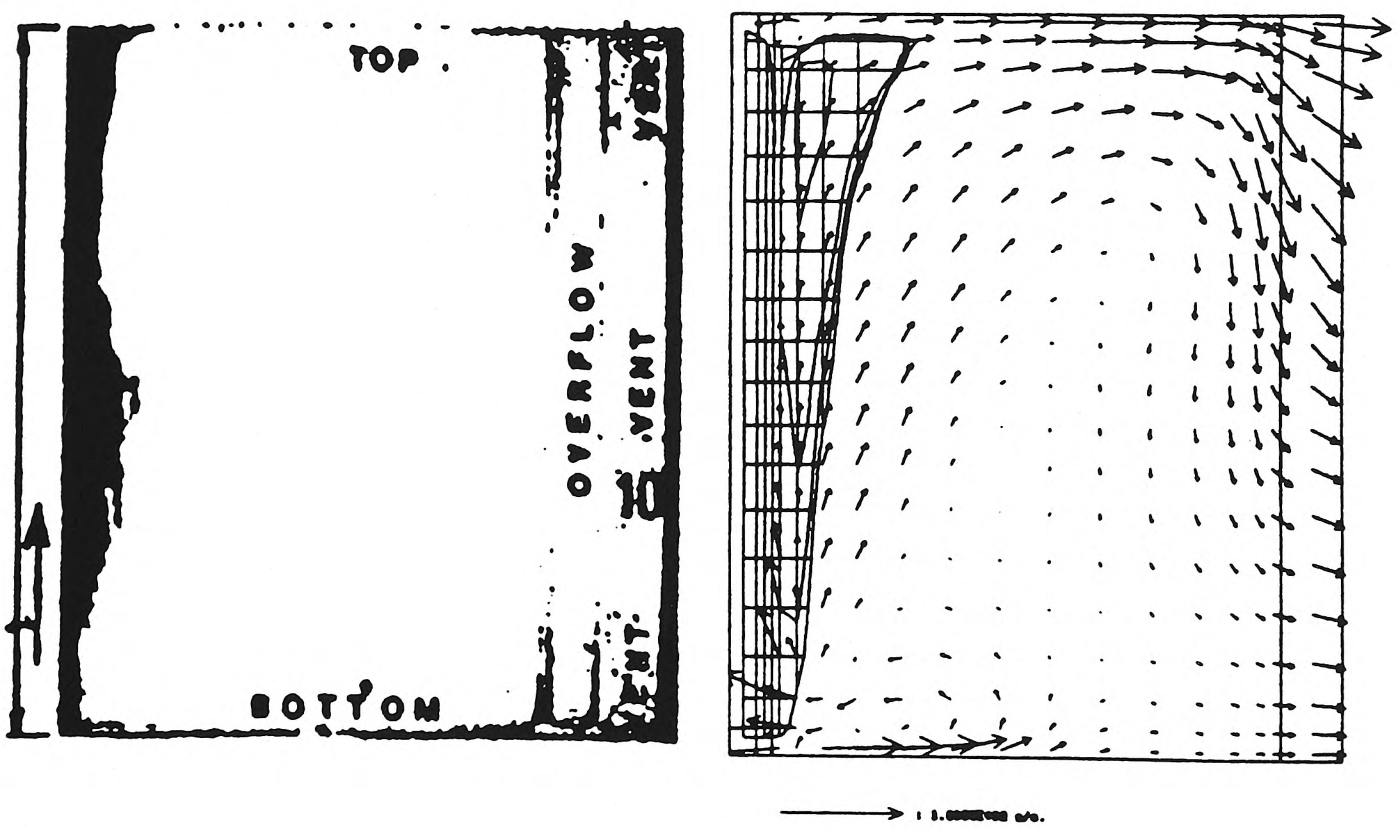


Figure 4.4 Comparison of the experimental and simulated results after 0.006 second.

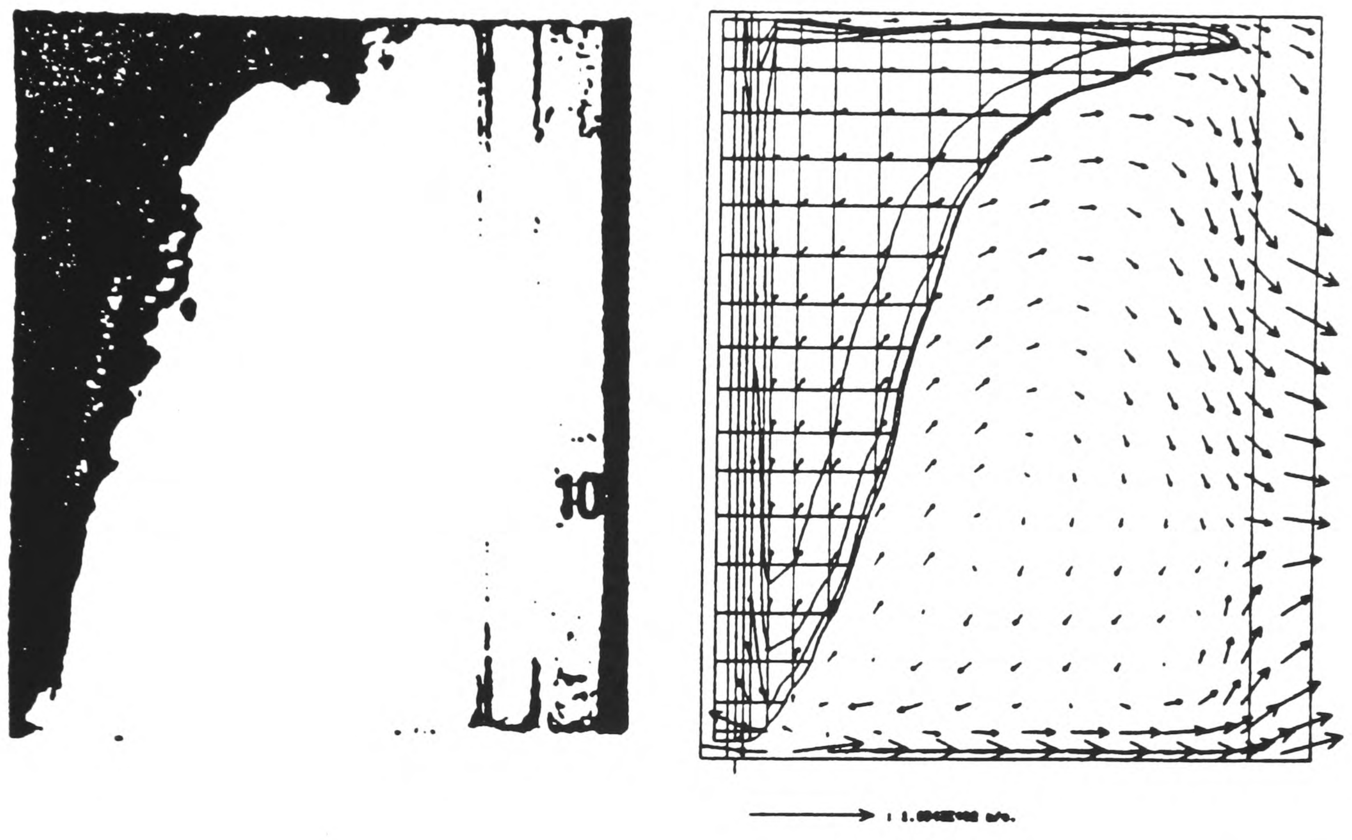


Figure 4.5 Comparison of the experimental and simulated results after 0.009 second.

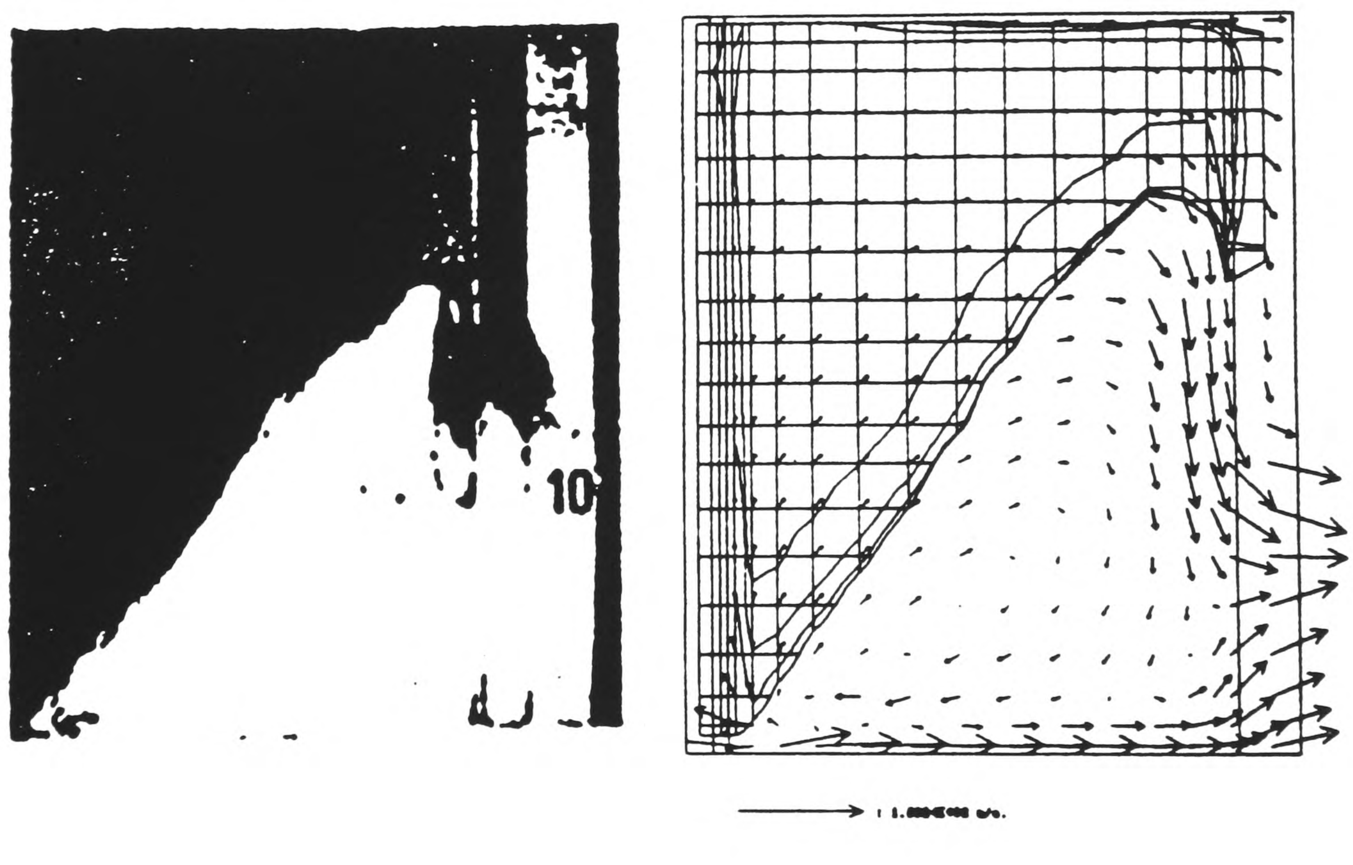


Figure 4.6 Comparison of the experimental and simulated results after 0.012 second.

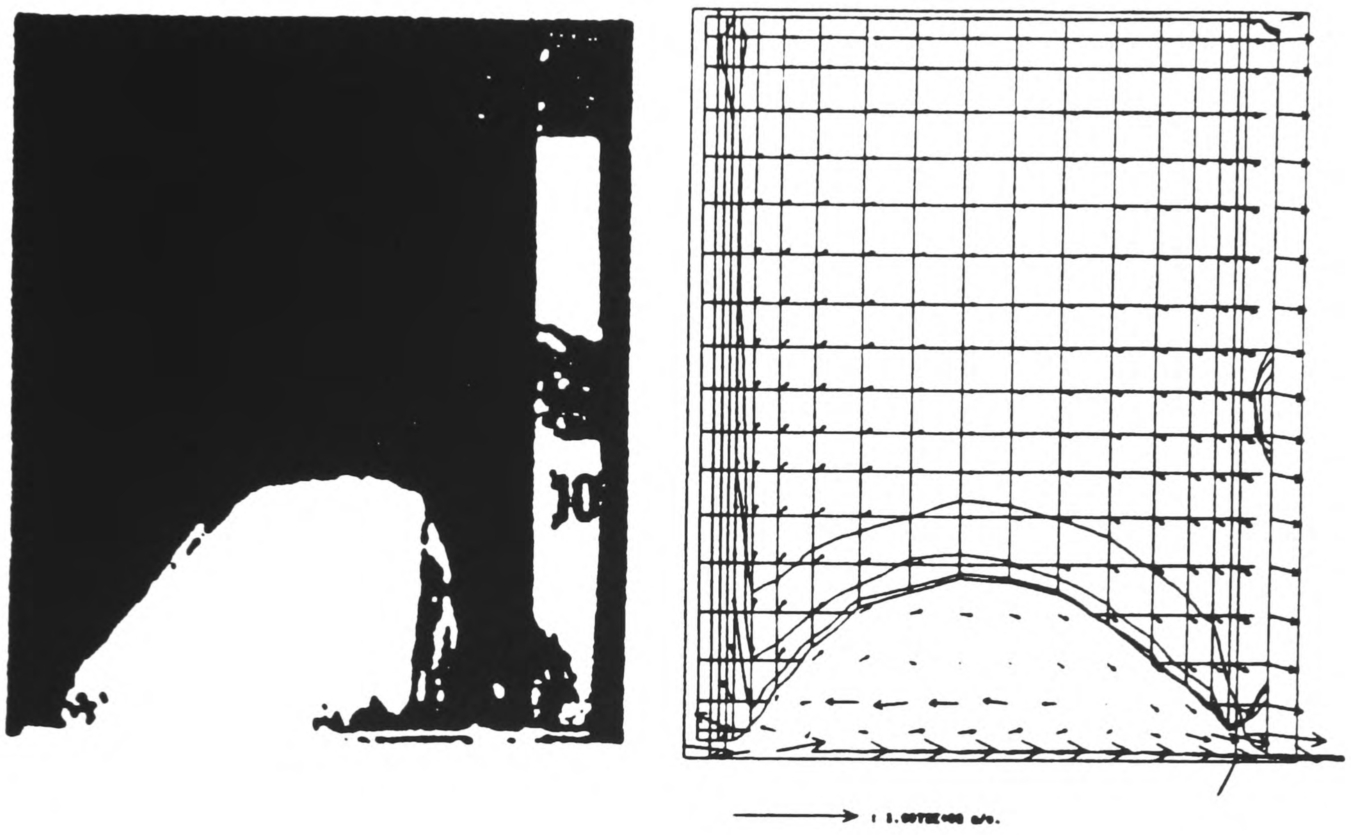


Figure 4.7 Comparison of the experimental and simulated results after 0.03 second.

4.2.4 Remarks for Validation Case I

An *effective kinematic viscosity* ν_t, 100 times that of the nominal kinematic viscosity ν, was used to simulate turbulence. A more sophisticated turbulence model such as κ-ε two-equation model [Harlow and Nakayama (1967)] may improve the results. Turbulence modelling, air compressibility, correct hydraulic losses in the runner and cavity due to friction, and variation in inflow velocity could have been included in the simulation to get more accurate predictions. However, this would inevitably increase the amount of CPU time needed by the model to simulate this situation. But, nevertheless, the simulated results are in good agreement with those obtained from experiments.

4.3 Validation case II: Sand Mould Filling with Aluminium

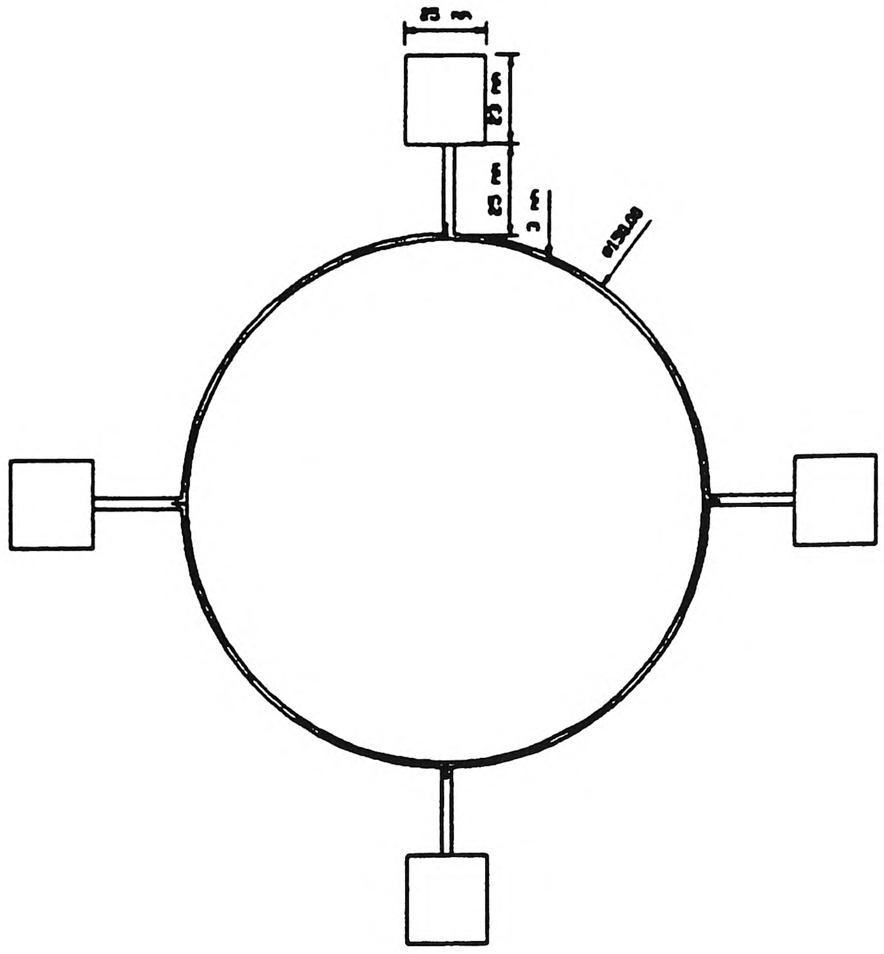
In 1991, an experiment was carried out by B Denton of BNF-Fulmer Metals Technology, Oxford, to investigate the feeding effect of vertical runners/raisers when casting a cylindrical vessel [Denton (1991)].

4.3.1 Problem Specification

The actual geometry of the cylindrical vessel is depicted in *Figure 4.8* (with top and front views shown). An aluminium alloy (A356) was used as the filling fluid and sand was used to make the mould and running system. The metal fluid was fed from the bottom of the four runners/raisers and then through their thin sections (3 mm thick) into the cylindrical mould cavity. Electrical transducers were embedded at various locations along the walls of the cylinder and runners to record the levels of metal liquid as the mould was being filled. The mould was cast at the temperature of 750 degree C. The fluid dynamic properties of the aluminium alloy (A356) is given in *Table 4.4.*

4.3.2 Computational Details

Since each of the four runners was positioned 90 degrees to its adjacent neighbours, in other words, the runners were set in symmetry, it was possible for the author to model only a section rather than the entire cylindrical vessel to obtain the same results. The originally curved section was then 'rolled' over to form a plate with a single runner/raiser attached to it as shown in *Figure 4.9*. Such that the problem could be



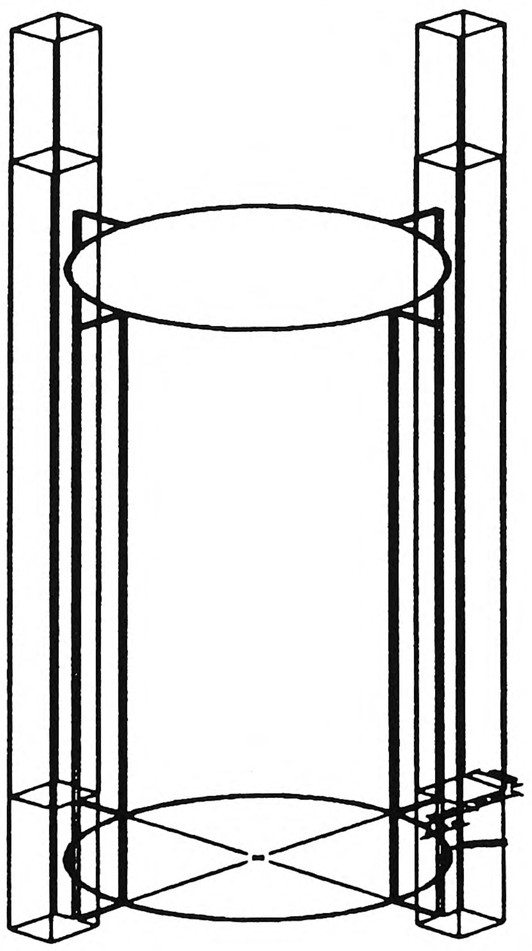


Figure 4.8 Schematic diagram of the running system and mould for the experiment.

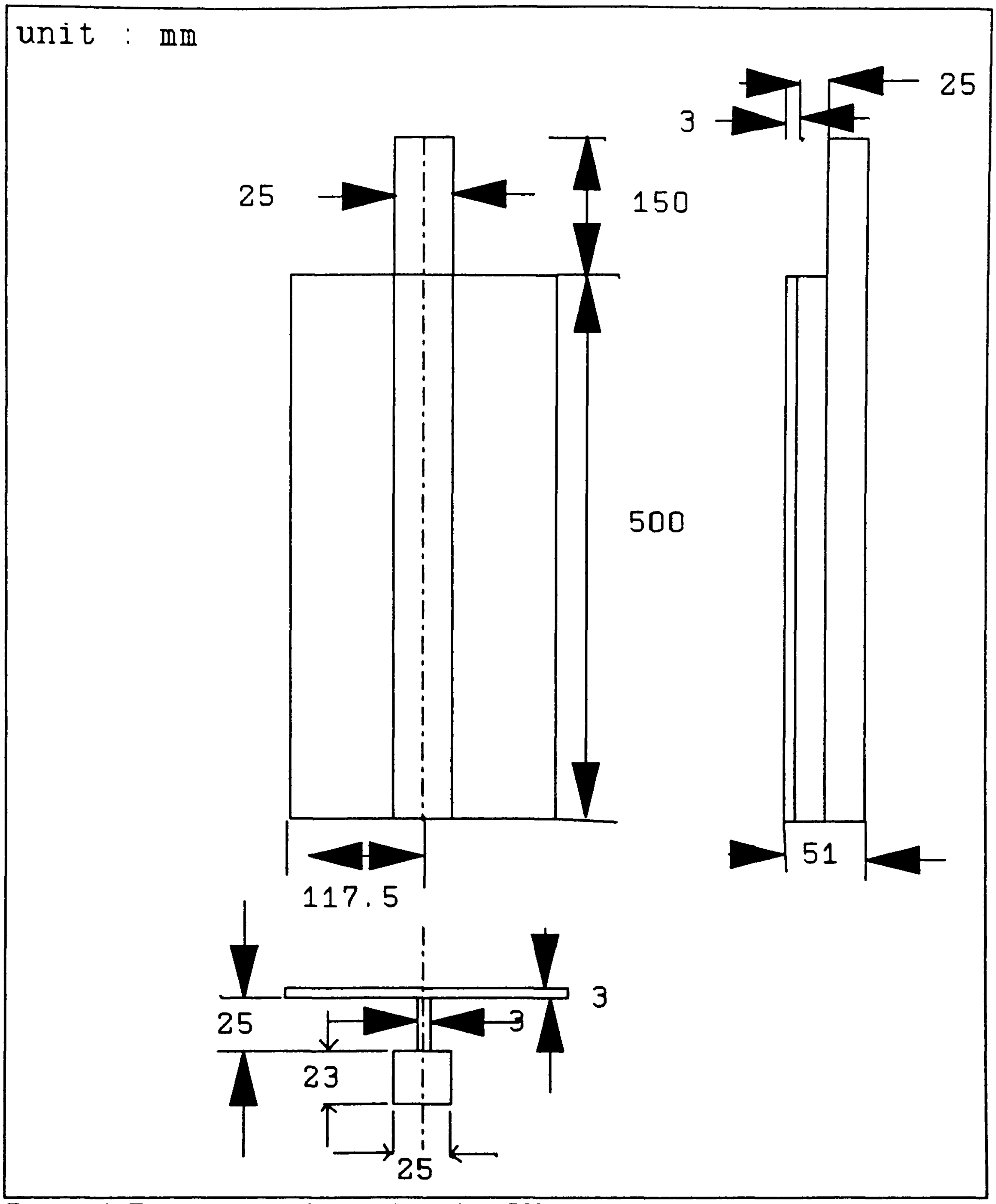


Figure 4.9 The geometry and dimensions of the BNF sand mould in details.

	Aluminium Alloy (A356)	Air
Density, ρ (Kg/m^3)	2420.0	0.99
Kinematic Viscosity, ν (m^2/s)	4.34x10^{-7}	1.417x10^{-5}

Table 4.4 Material properties for validation case II.

simulated with cartesian coordinates. To simplify matters further, the plate was subdivided into two parts along a symmetry plane which lied at the middle of the runner, see *Figure 4.9* for detail.

A mesh of 6 x 12 x 7 = 504 cells in three-dimensions was used to model the problem. The mesh, with blocked cells not shown, can be found in *Figure 4.10.* The initial and boundary conditions are listed in *Table 4.5.* A real time of 5 seconds was simulated which took approximately 46 wall-clock hours to complete when running on a Norsk Data ND5900. A total of 3500 time-steps with δt varied from 0.001 to 0.005 second in a series of continuation runs were required to achieved the results. The rest of the run-time parameters are listed in *Table 4.6.*

4.3.3 Results

Figure 4.11 shows velocity vectors which indicate how the air is moving along the runner/raiser and within the mould cavity before the molten metal flows in.

Figures 4.12 and *4.13* show the progress of the filling after 0.25, 0.5, 2, 3, and 5 seconds respectively.

The average height of metal in the runner obtained from the experiment and the simulation can be found in *Table 4.6* and its corresponding graph in *Figure 4.14.* Similarly, average height of metal in the mould cavity can be found in *Table 4.7* and *Figure 4.15* respectively.

Initial conditions:

Inflow velocity: 0.313 m/s

Outlet (at top of runner): Pressure at outlet boundary is prescribed to zero.

Boundary conditions:

Walls: impermeable, constant temperature, and non-slip or zero velocity at walls is assumed.

Gravity acting downward: 9.81 ms^{-2}

Other information:

Liquid volume flow rate : $8.99x10^{-5}$ m^3/s

Volume of computational domain: $3.82x10^{-4}$ m^3

Expected fill-up time : 4.2 s

Table 4.5 Initial and boundary conditions for validation case II.

Relaxation factors:

Linear: pressure (0.9)

False time-step: u (0.3), v (0.1), w (0.01)

Time-step size varies from 0.001 to 0.005

Residual tolerance for velocities and pressure: 10^{-6}

Number of sweep per time-step = 70

Total number of time-steps executed: 3500

Simulated real time: 5 seconds

Total run-time (wall clock) = 46 hours (Norsk Data ND5900)

Table 4.6 Run-time parameters for validation case II.

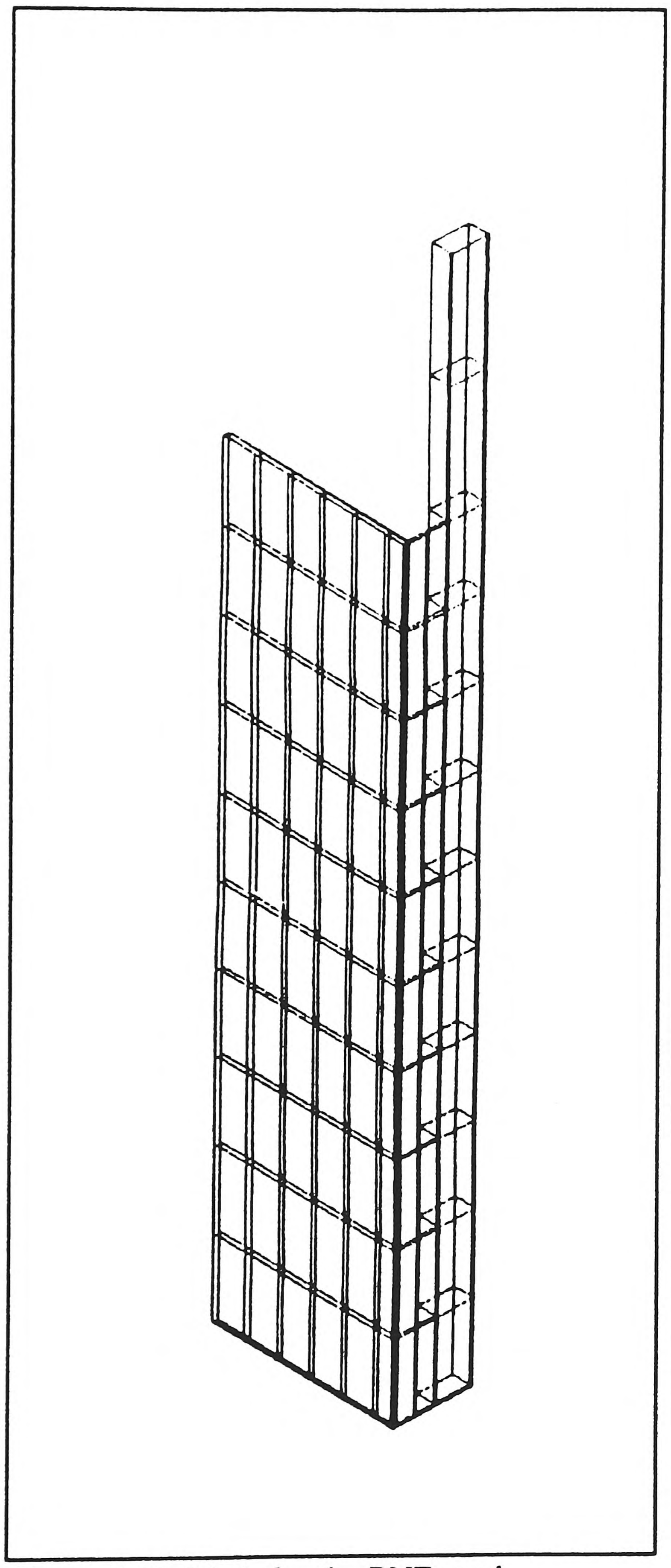


Figure 4.10 Mesh for the BNF sand mould (blocked cells are not shown).

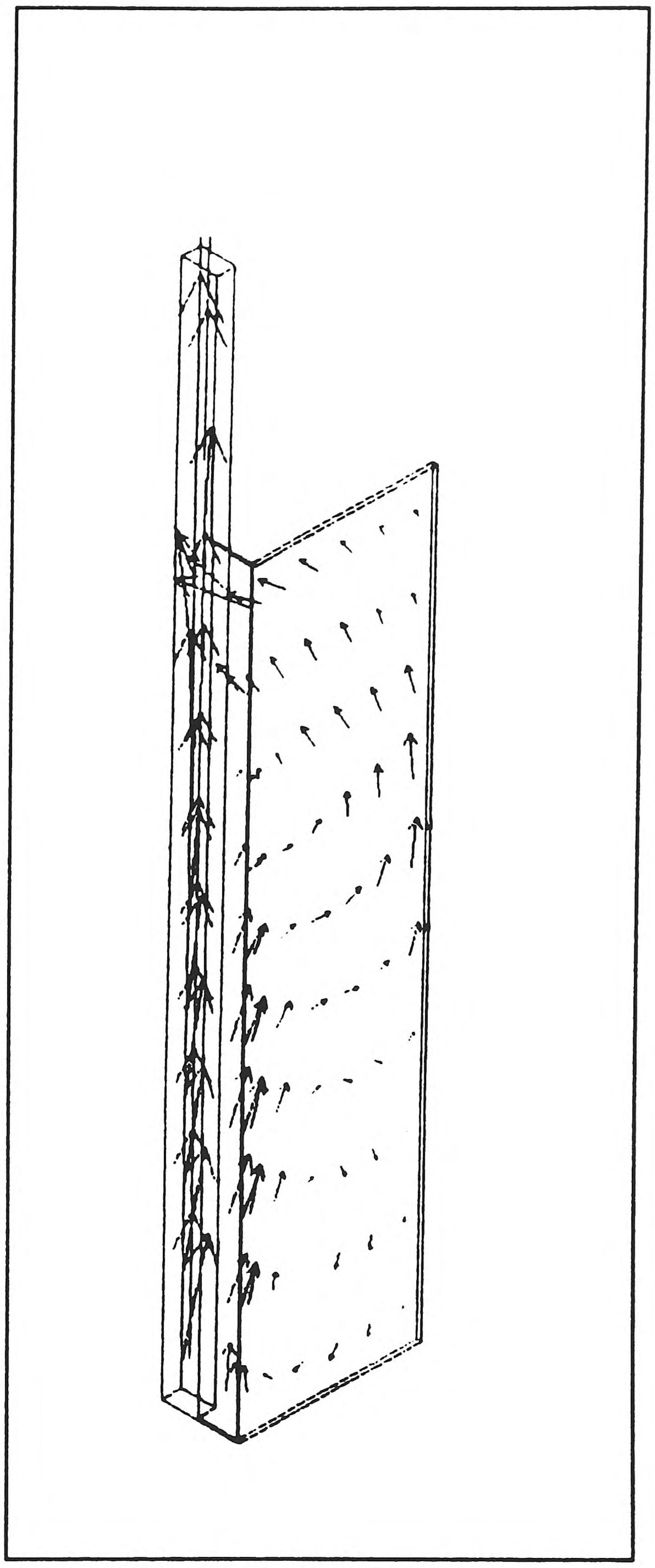


Figure 4.11 Velocity vectors illustrate the direction of filling inside the BNF sand mould.

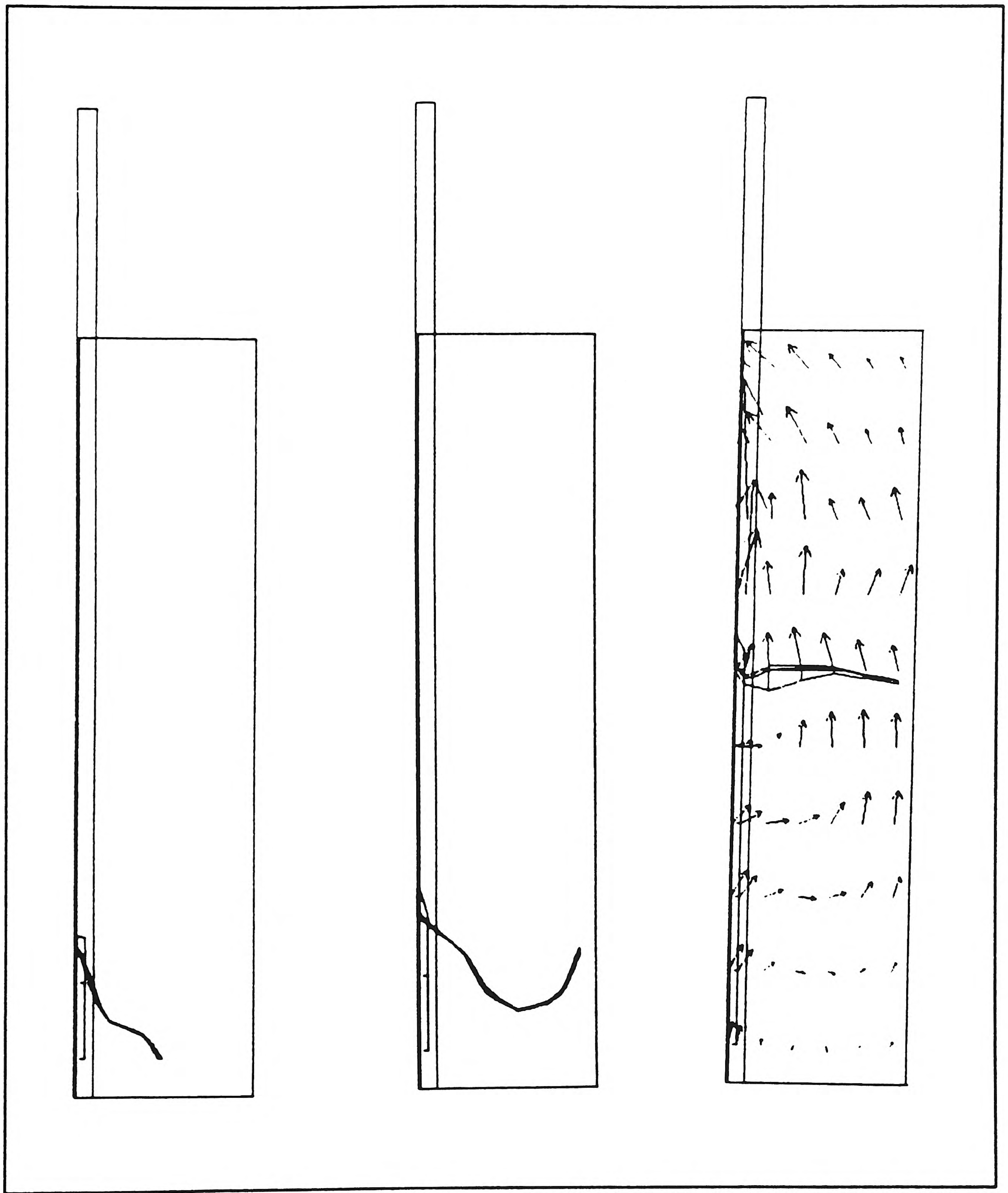


Figure 4.12 Liquid free surface levels in the sand mould (from left to right) after 0.25, 0.5 and 2 seconds.

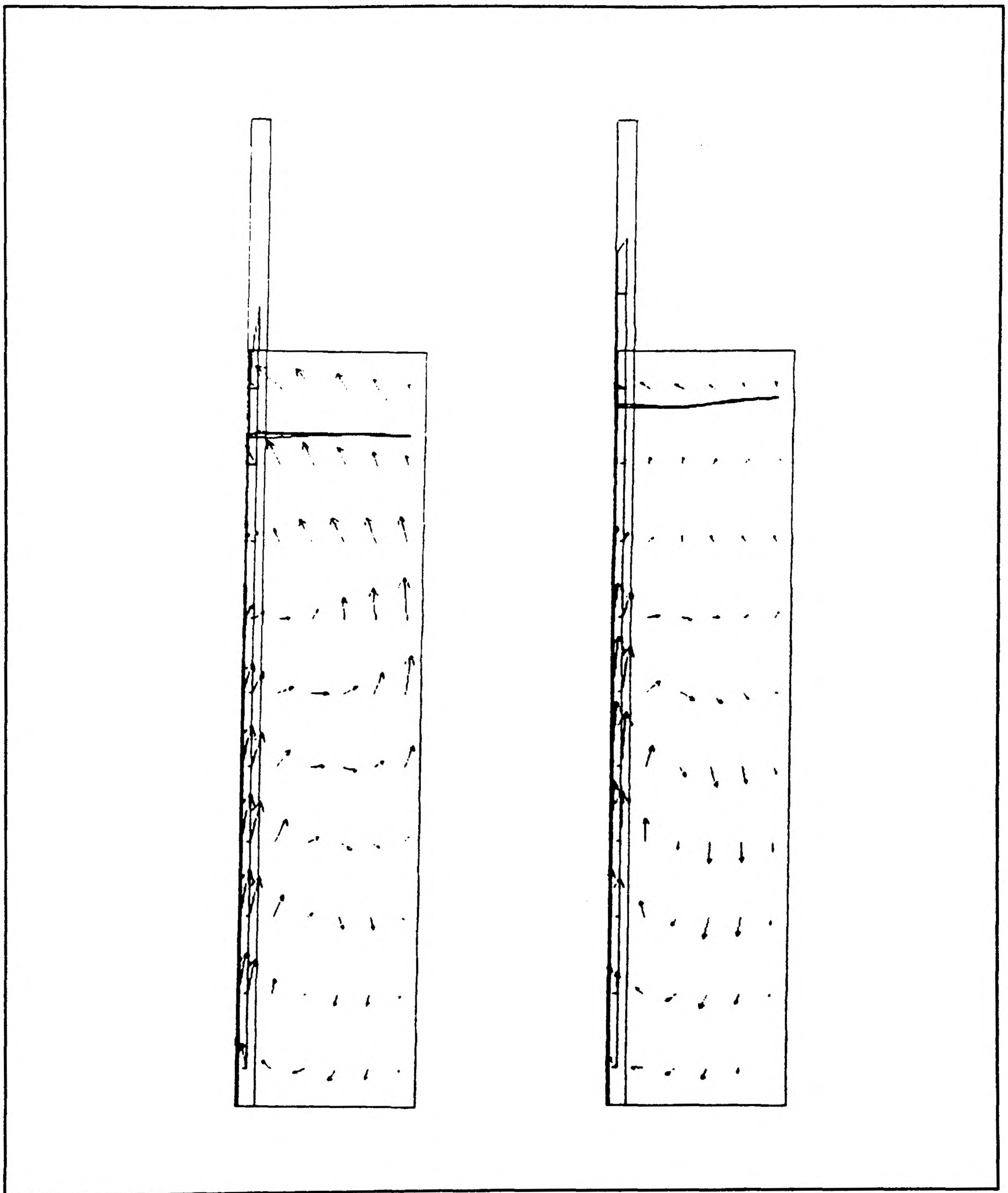


Figure 4.13 Liquid free surface levels in the sand mould (from left to right) after 3 and 5 seconds.

Time (s)	Average height of metal in runner - Experiment (mm)	Average height of metal in runner - Simulation (mm)
0.03	9	7
0.18	48	46.5
0.38	88	63.9
0.83	128	111
1.18	168	157.4
1.61	208	214
2.09	265	278.7
2.81	328	375
3.23	387	434
3.92	444	507

Table 4.6 The experimental and simulated results for the average height of liquid metal front in the runner.

Time (s)	Average height of metal in casting - Experiment (mm)	Average height of metal in casting - Simulation (mm)
0.26	18	39.5
0.46	53	65.04
0.83	94	106.2
1.29	140	160.3
1.55	175	195
1.95	217	243
2.43	250	301.2
2.92	319	364
3.58	371	398
4.12	426	408

Table 4.7 The experimental and simulated results for the average height of liquid metal front in the mould cavity.

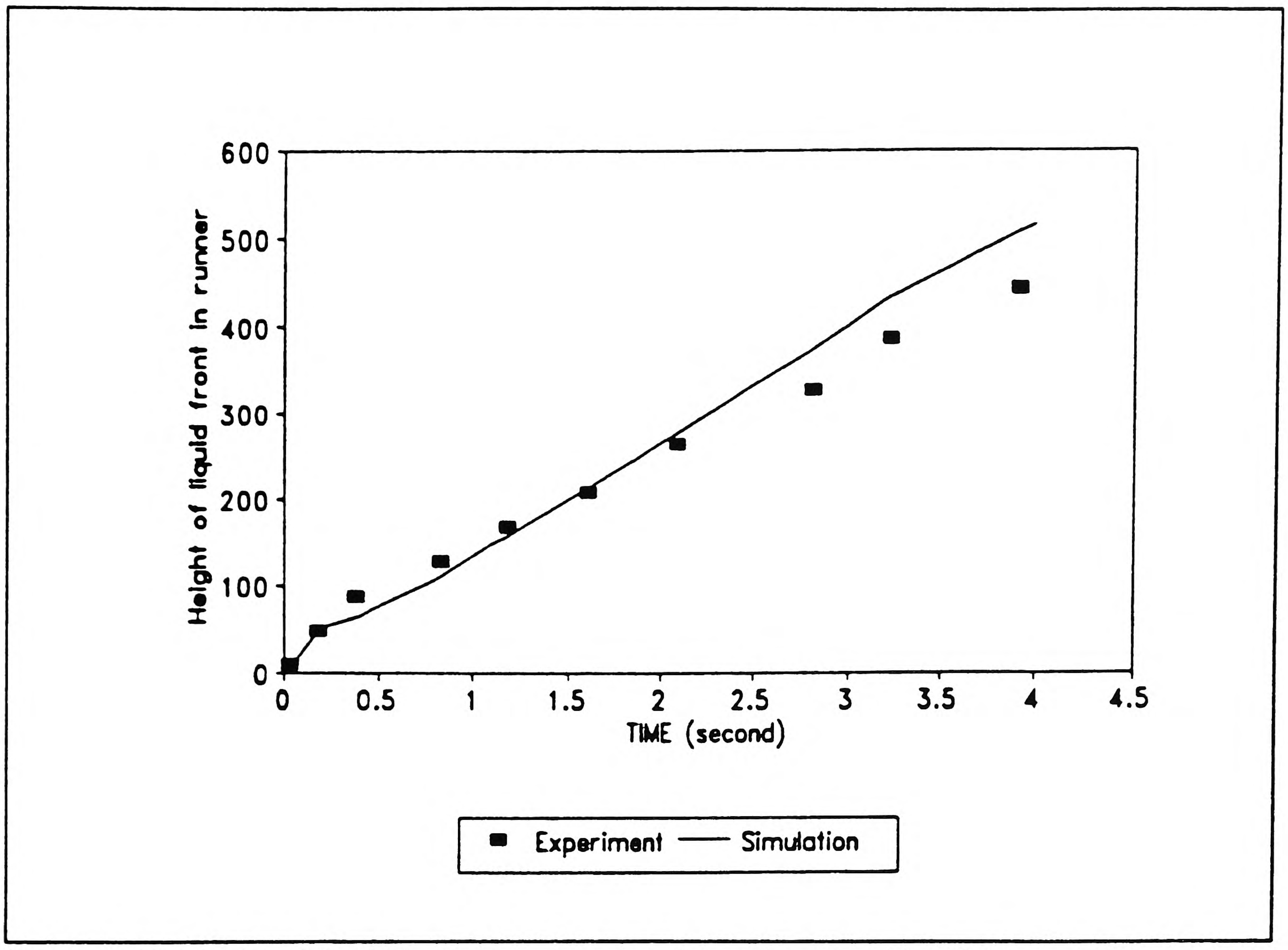


Figure 4.14 Height of liquid in the runner - experimental vs simulated data

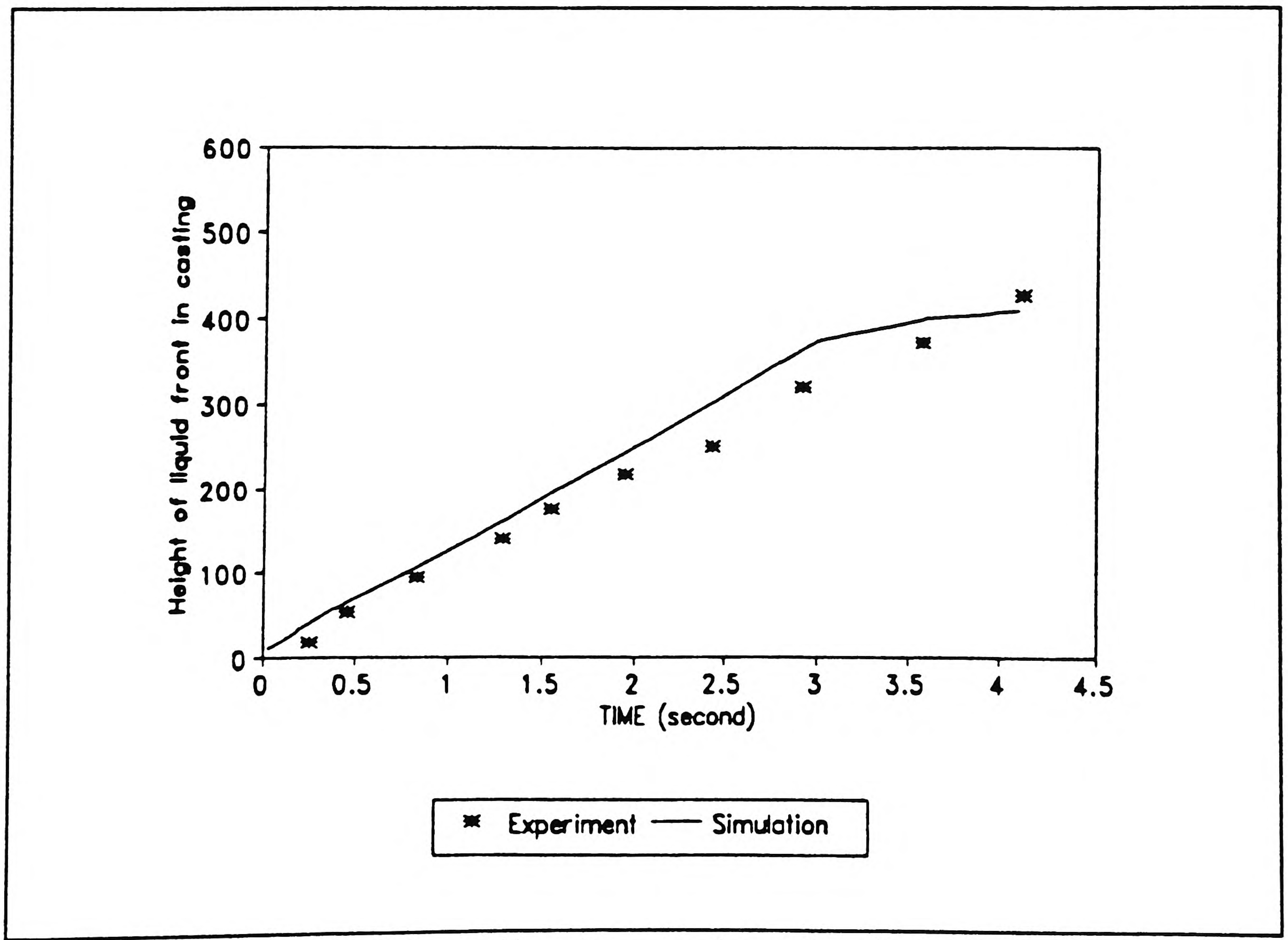


Figure 4.15 Height of liquid in mould cavity - experimental vs simulated data.

4.3.4 Remarks for Validation Case II

The differences between the experimental results and the simulation may be due to the following:

1. The effect of both heat transfer and solidification were not simulated. Therefore, the effect of liquid density and viscosity varying with temperature was not taken into account. Also, the effect of surface tension which becomes important when the surface of the aluminium alloy solidifies due to contact with the cold sand walls was not included.
2. The pressure driving the inflow velocity varies in the experiment due to the weight of the metal inside the mould. However, it was kept constant during the simulation.
3. Although numerical smearing at the free surface was reduced by the van Leer scheme, it may still become excessive when the flow was not smooth, especially when the flow had to move around a tight corner (see *Figure 4.11*). This problem may be alleviated with a finer mesh and smaller time-step size but with a heavy cost of computing time.
4. With a hindsight, a degree of inaccuracy might be inherent in the model due to the use of a rectangular mesh. The use of polar-cylindrical mesh to model the entire mould might yield better results.
5. Further possible explanation for the difference could be the accuracy of the data collected by the equipment at BNF Metals Technology and their interpretation.

4.4 Industrial Case I: 3-D Filling of Part of a Gating System

This example demonstrates the capability of the SEA filling algorithm in capturing the formation of excessive waves. These could result in the possible air entrainment or voids within the casting during filling . The geometry of the example is part of the gating system attached to a rectangular mould cavity as shown in *Figure 4.16*. The data of the geometry was provided by BNF-Fulmer in 1990 [Denton (1990)]. Steel was used as the material for filling.

4.4.1 Problem Specification and Computational Details

Figure 4.17 gives detailed dimensions of the geometry of the example under

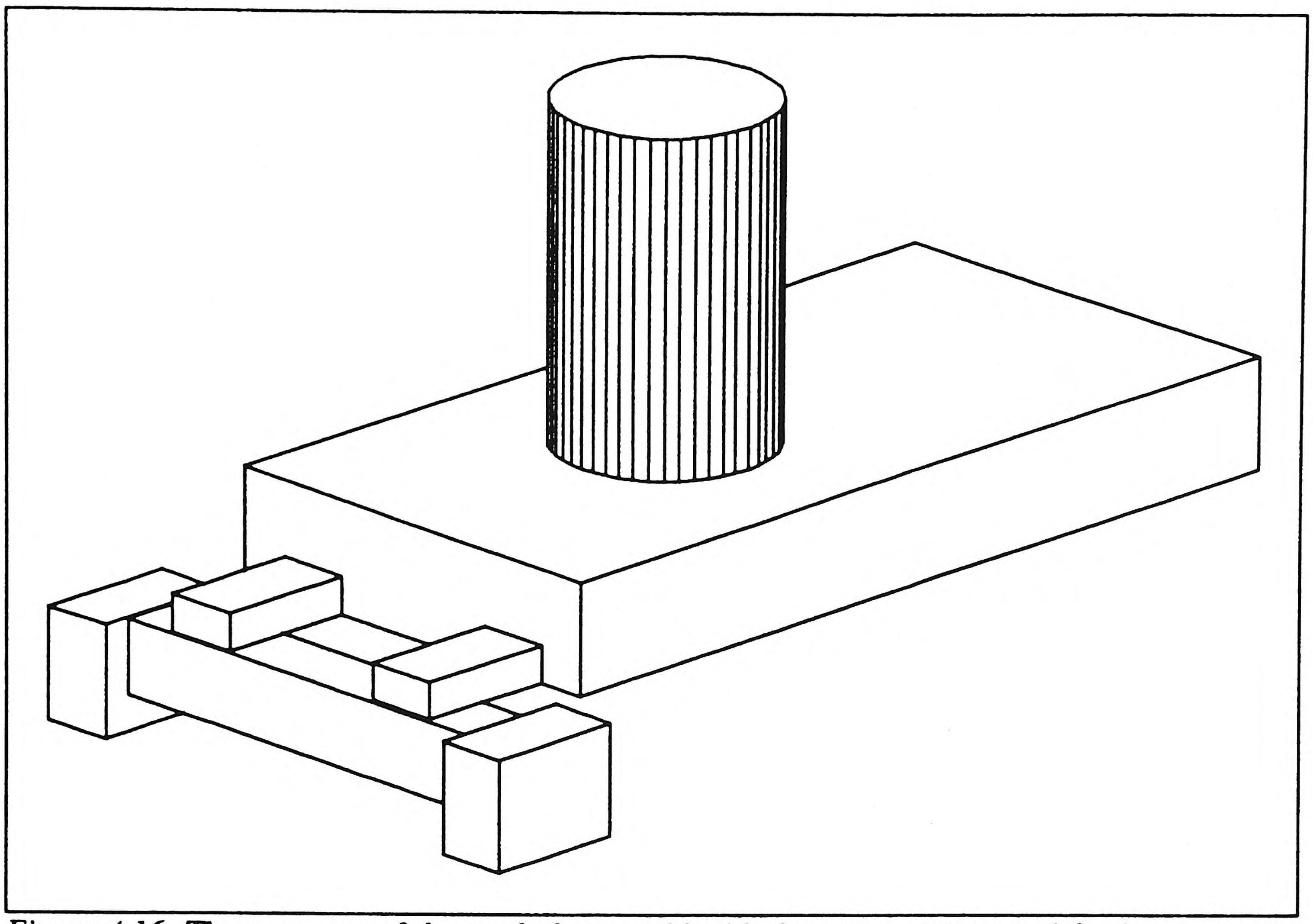


Figure 4.16 The geometry of the steel plate mould with the gating system and feeder shown.

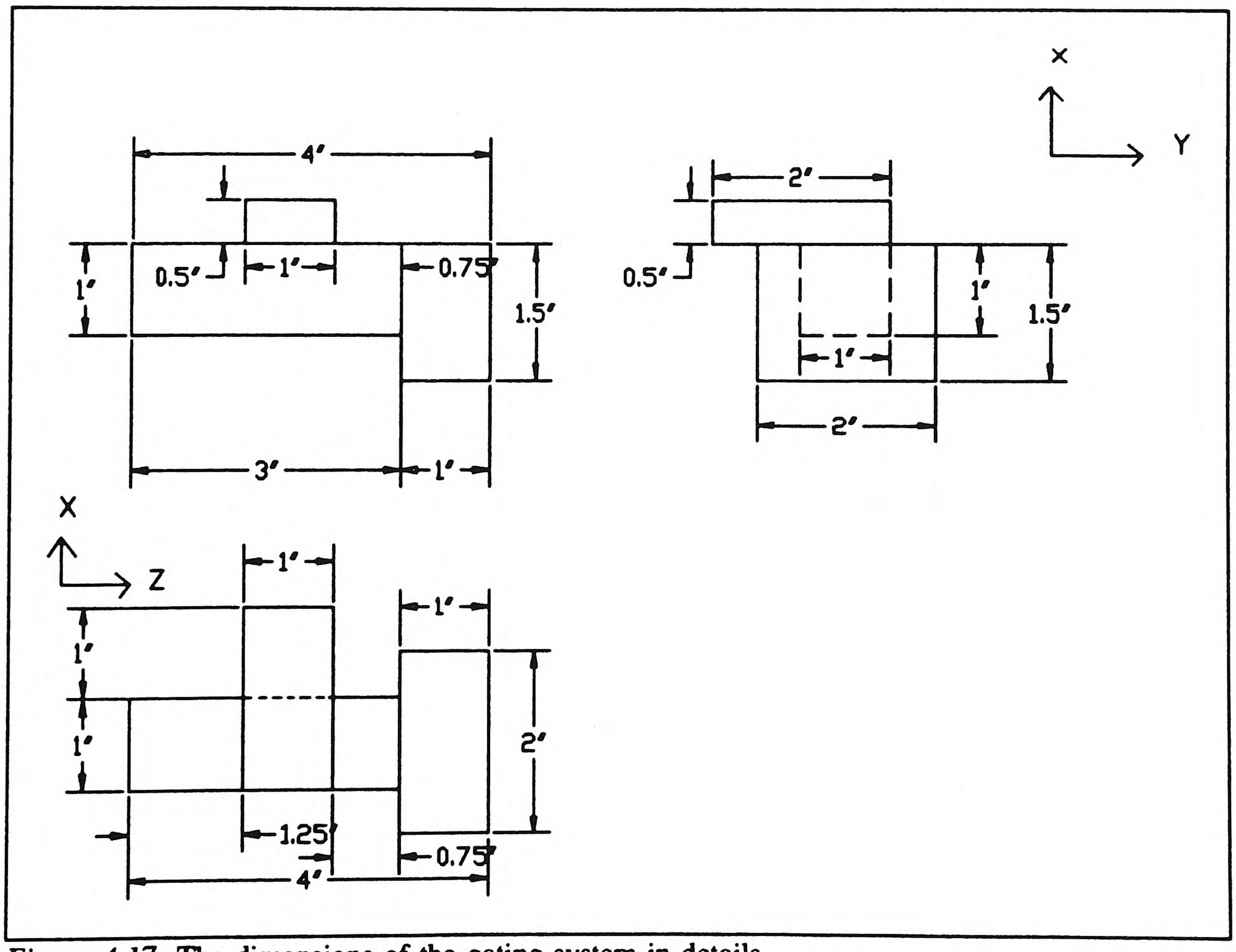


Figure 4.17 The dimensions of the gating system in details.

investigation. *Figure 4.18* shows the geometry of the gating system used in the simulation. Although, a three-dimensional structured grid of 12 x 11 x 17 = 2244 cells (see *Figure 4.19*, in which the blocked cells are not shown) and a coarser grid of 10 x 8 x 15 = 1200 cells were used to model the problem, only the results from the finer grid will be shown here. The results from both the coarser and finer grids are very similar, except that more details can be seen with the finer grid [Chan K S et al (1991)]. The material properties for the simulation are listed in *Table 4.8*.

At the initial stage of filling, metal fluid was poured into the running system (at point A in *Figure 4.18*) presumably via the sprue (not shown and not modelled) under gravity. The inflow velocity was set to 1.868 m/s.

	Steel	Air
Density, ρ (Kg/m^3)	7560.0	1.0
Kinematic Viscosity, ν (m^2/s)	10^{-6}	10^{-5}

Table 4.8 Material properties for the industrial case I.

The fluid would then fill up the runner, 'dross trap' and ingate before reaching the main mould cavity system at point B in *Figure 4.18*. At the same time, the air which existed within the cavity originally would then be expelled from the running system via the exit at point B. The initial and boundary conditions used in the simulation are summarised in *Table 4.9*.

For the finer grid of 2244 cells, time step of 0.001 second was chosen to ensure the stability under the Courant criterion (*Equation 3.32, Chapter 3*). At least 25 sweeps per time step were needed to ensure convergence. A total of 760 time steps were required to almost fill the chamber of the example. This represents a real time simulation of 0.76 second. The rest of the run time parameters are listed in *Table 4.10*.

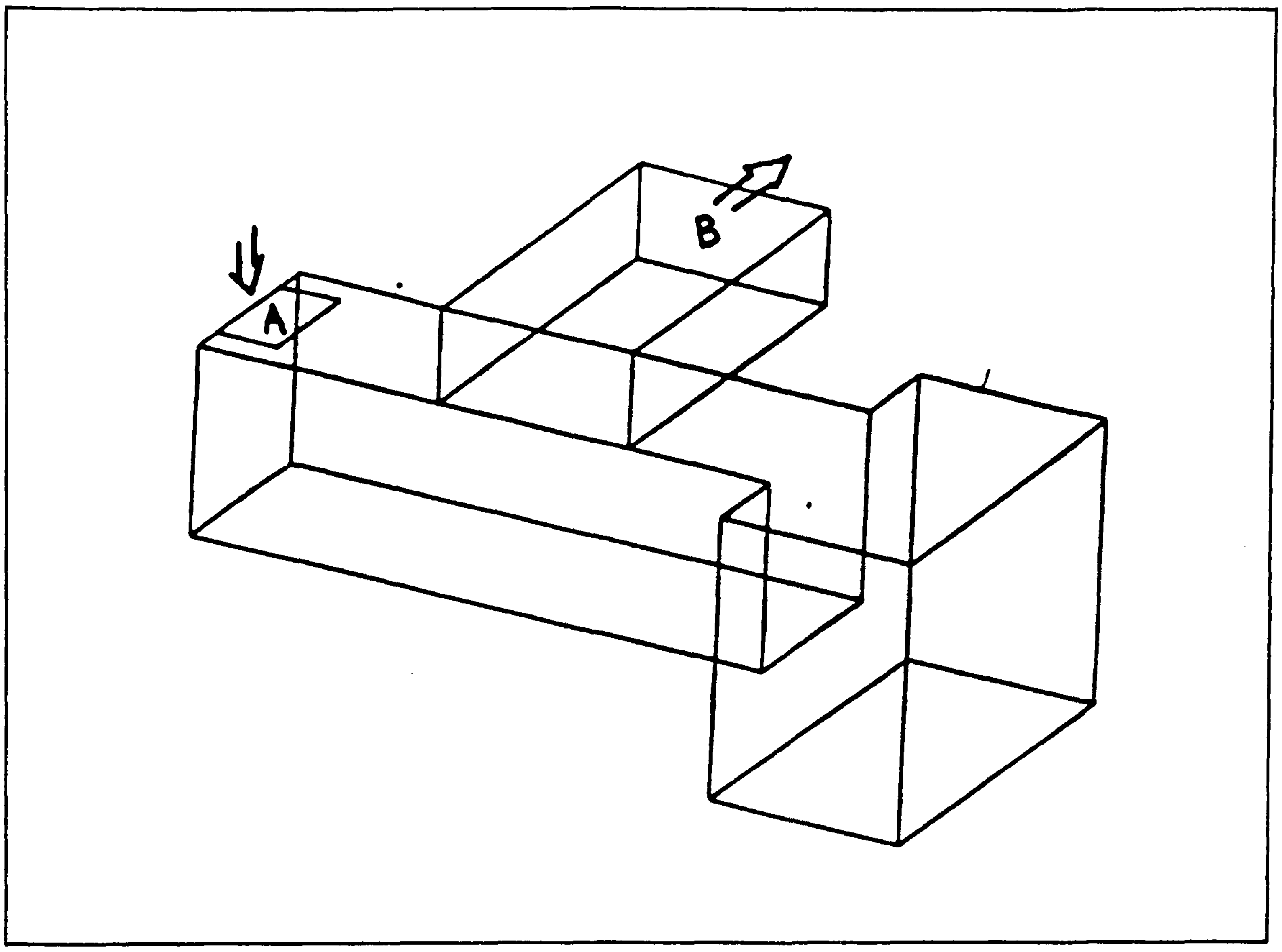


Figure 4.18 Sand mould geometry of the gating system.

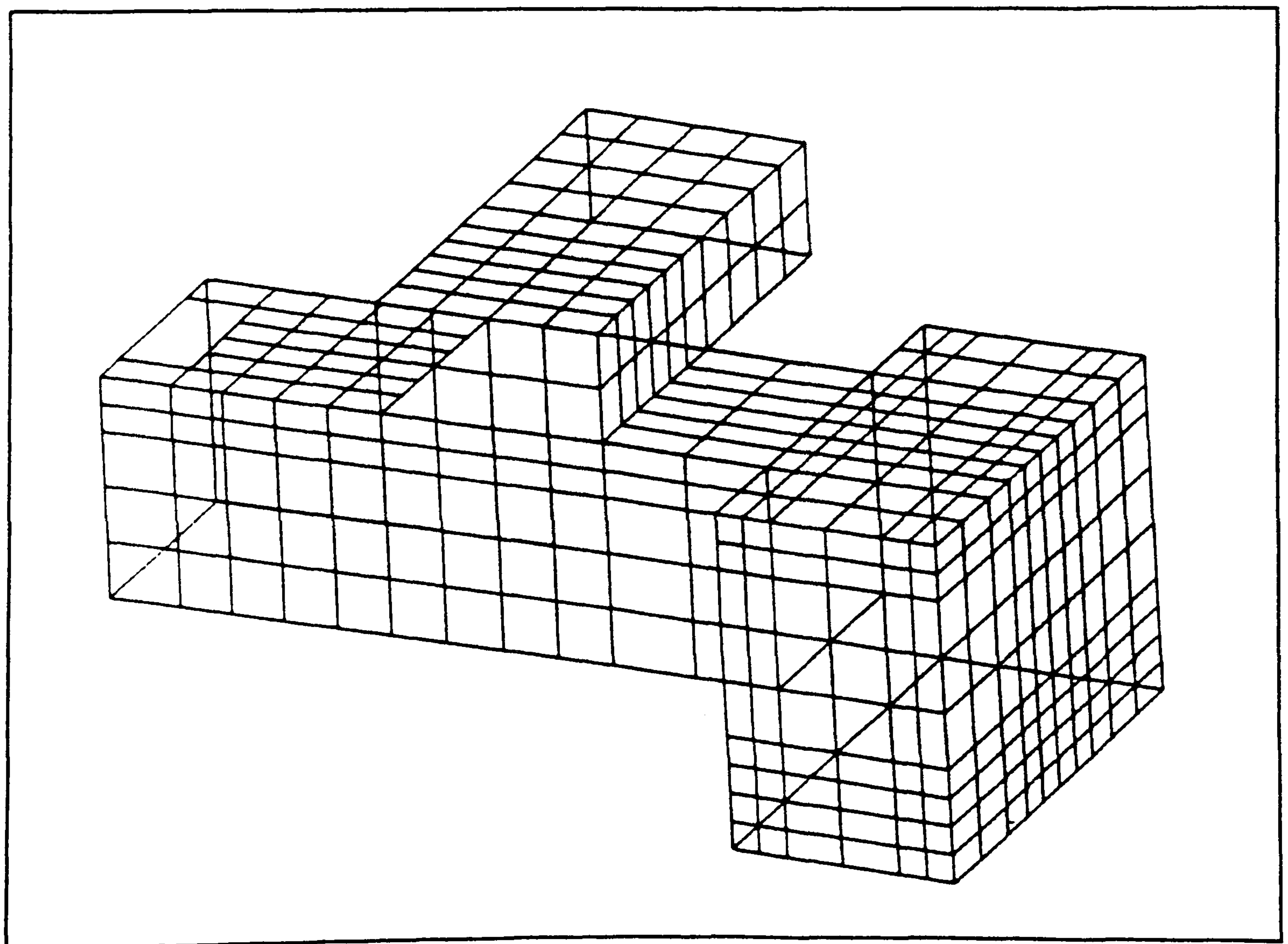


Figure 4.19 3-D computational grid for the gating system.

Initial conditions:

Inflow velocity (enter the mould at point A): 1.868 m/s

Outlet (at point B): Pressure at outlet boundary is prescribed to zero.

Boundary conditions:

Walls: impermeable, isothermal, and non-slip or zero velocity at walls is assumed zero.

Gravity acting downward: 9.81 ms^{-2}

Other information:

Liquid volume flow rate : 2.35×10^{-4} m^3/s

Volume of mould cavity: 1.147×10^{-4} m^3

Expected fill-up time : 0.49 s

Table 4.9 Initial and boundary conditions for industrial case I.

Relaxation factors:

Linear: pressure (0.7)

False time-step: u(0.003), v(0.003), w(0.003)

Time-step size: 0.001 s

Residual tolerance for pressure and velocities: 10^{-6}

Number of sweeps per time step: varies from 25 to 40

Total number of time steps: 760

Simulated real time: 0.76 seconds

Total run-time (wall clock): 41 hours (Norsk Data ND5900)

Table 4.10 Run-time parameters for industrial case I.

4.4.2 Results

The results of the computation are presented graphically in *Figures 4.20* to *4.30* at selected time intervals. The advancement of the air/liquid interface is denoted by a isosurface at $\phi = 0.5$. In addition, selected vector plots indicate the flow pattern of both the air and liquid masses.

After 0.03 seconds of simulation in real time, the metal fluid entered the running system in the form of a liquid column, as shown in *Figure 4.20. Figure 4.21* shows the liquid jet nearly reaching the end wall of the dross trap after 0.09 second.

Figure 4.22 shows the subsequent jet of liquid impacting on the end wall of the dross trap and spreading after 0.15 second. *Figure 4.23* shows the free surface at a cross-section of the mould, together with the flow field vectors. The subsequent spreading of the liquid surface as the result of the jet impingement upon the wall can clearly be seen. Note that a recirculation cell or air void is developing below the jet as air is being trapped. Above the liquid surface, air initially follows the liquid movement until it reaches the end wall. There it reverses and flows along the top of the runner towards the ingate (ie. the outlet).

Figure 4.24 (after 0.18 second) shows the advancement of the liquid surface at the end of the dross trap as it first flows along the base wall, forming a third interface. *Figures 4.25* and *4.26* (after 0.21 and 0.24 second) show the lower layer of the liquid surface closing upon itself in a complex manner, trapping a quantity of air within it. In addition, a wave is developing at the top layer of the liquid surface which is moving away from the end wall towards the outlet. As it approaches the outlet, it forms a crest (*Figure 4.26*), which is amplified by fast-flowing air that is trying to squeeze past it as it escapes through the outlet. The consequence of the wave is that liquid is escaping towards the outlet before the dross trap is full. Hence, the effectiveness of the dross trap to retain impurity and to stable the liquid flow is compromised.

Figures 4.28 (after 0.33 second) to *4.30* (after 0.75 second) show the dross trap and the rest of the runner gradually being filled up. Note the diminution of the trapped air void or bubble is shown in *Figure 4.29*. *Figure 4.30* shows that the entire running system is nearly full and the air void in the dross trap virtually disappeared. However, the liquid escapes prematurely due to the wave action causing the filling process to take longer than expected (ie. expected fill time is 0.49 seconds).

4.4.3 Remarks for Industrial Case I

Excessive wave and air entrapment in the gating system would certainly mean the that quality of the liquid being fed into the main mould cavity would become undesirable. Thus, poor castings would result. Presume that the simulated results predicted by the SEA method for this scenario were 'correct' then the requirement of having the dross trap at the end of the runner dose not seem to be justified.

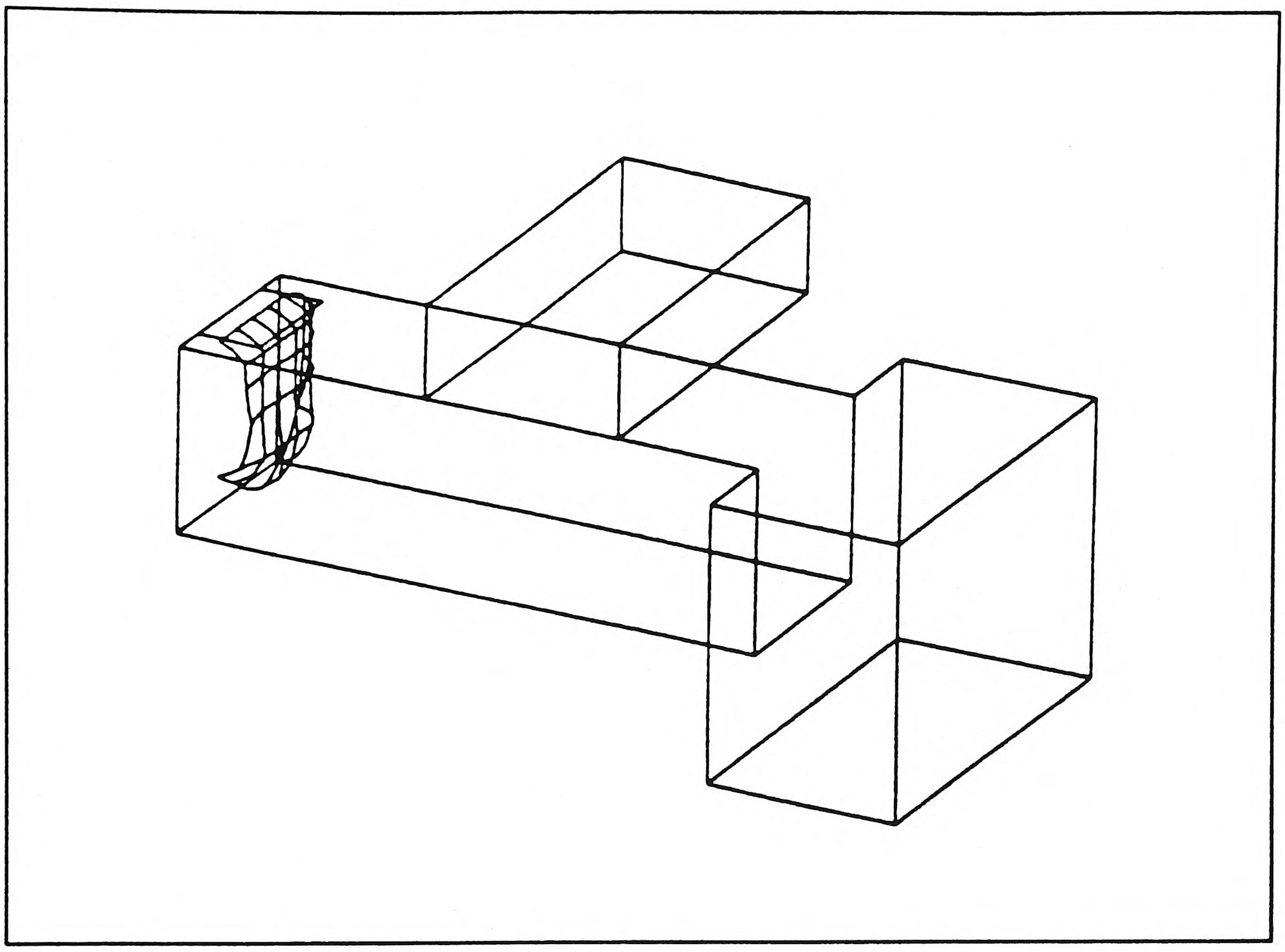


Figure 4.20 Liquid surface after 0.03 second.

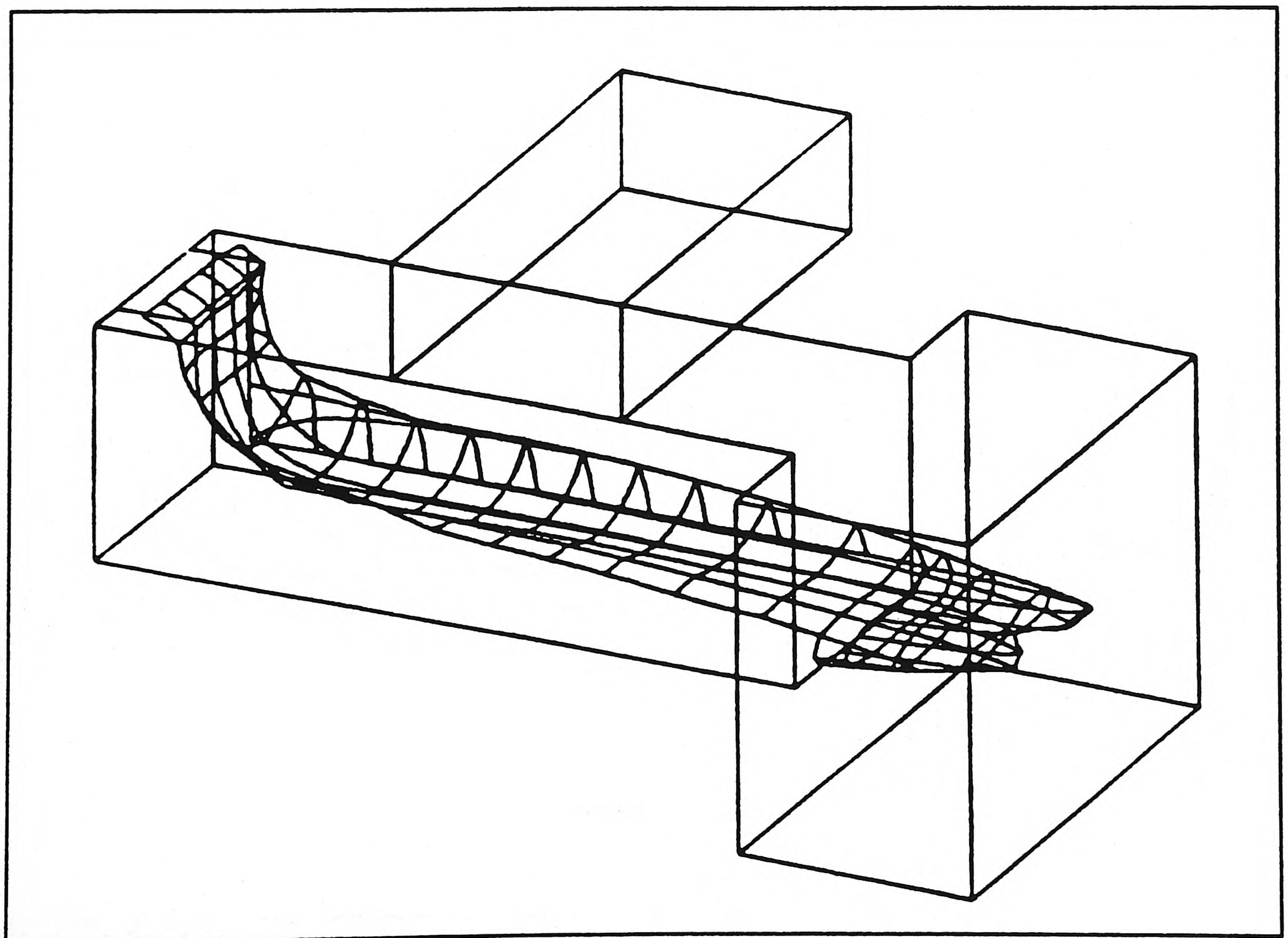


Figure 4.21 Liquid surface after 0.09 second.

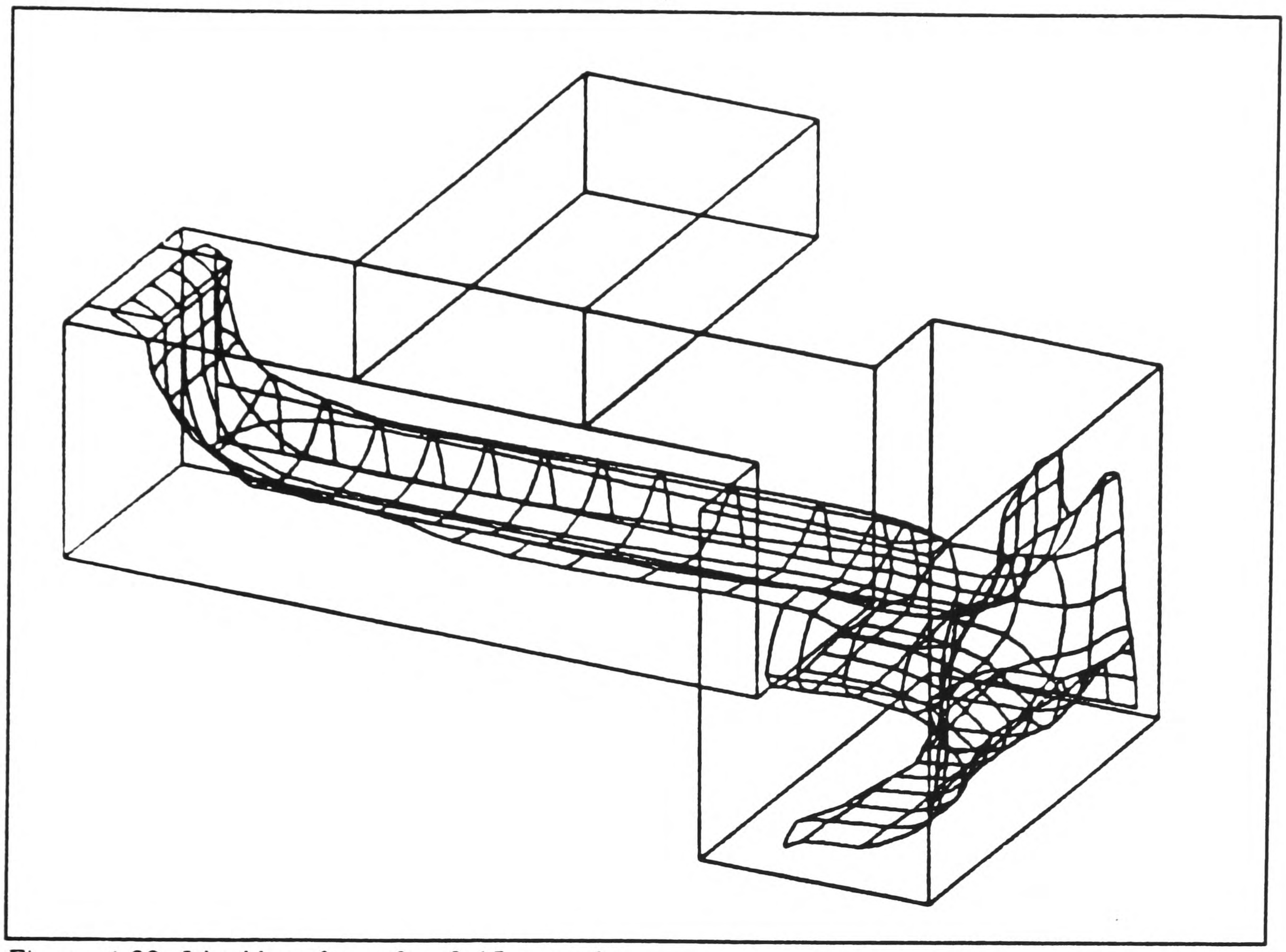


Figure 4.22 Liquid surface after 0.15 second.

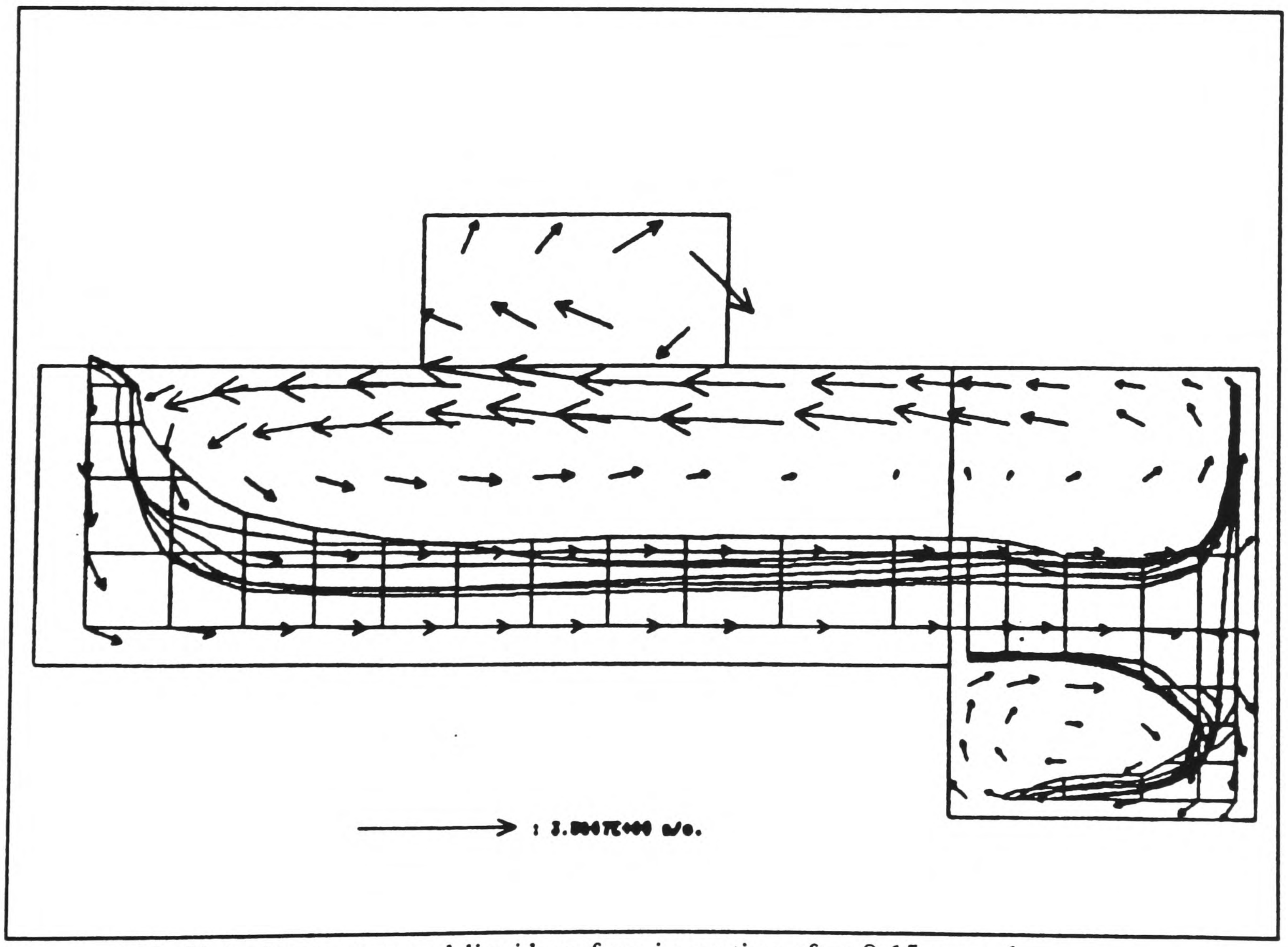


Figure 4.23 Velocity vectors and liquid surface in section after 0.15 second.

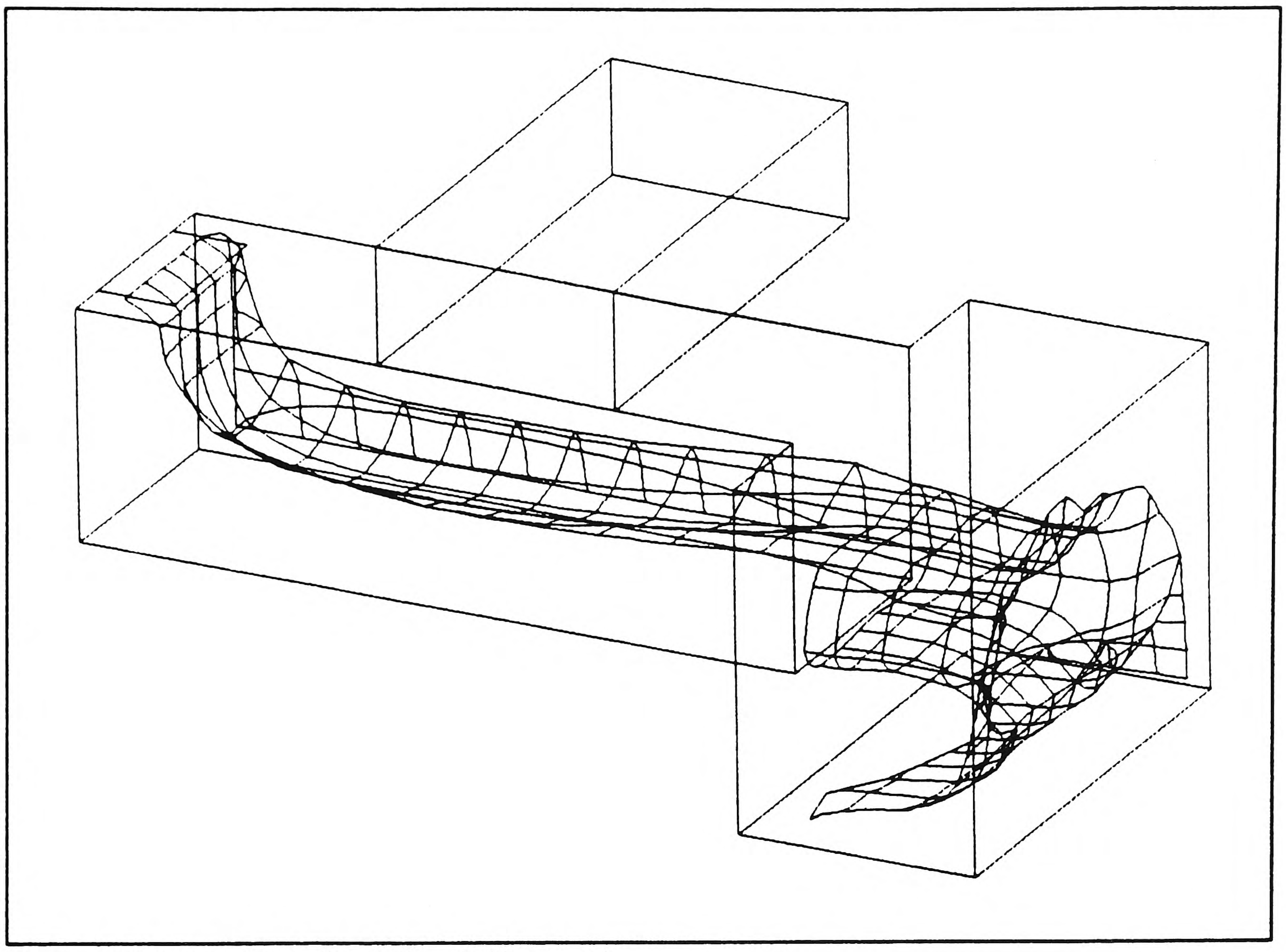


Figure 4.24 Liquid surface after 0.18 second.

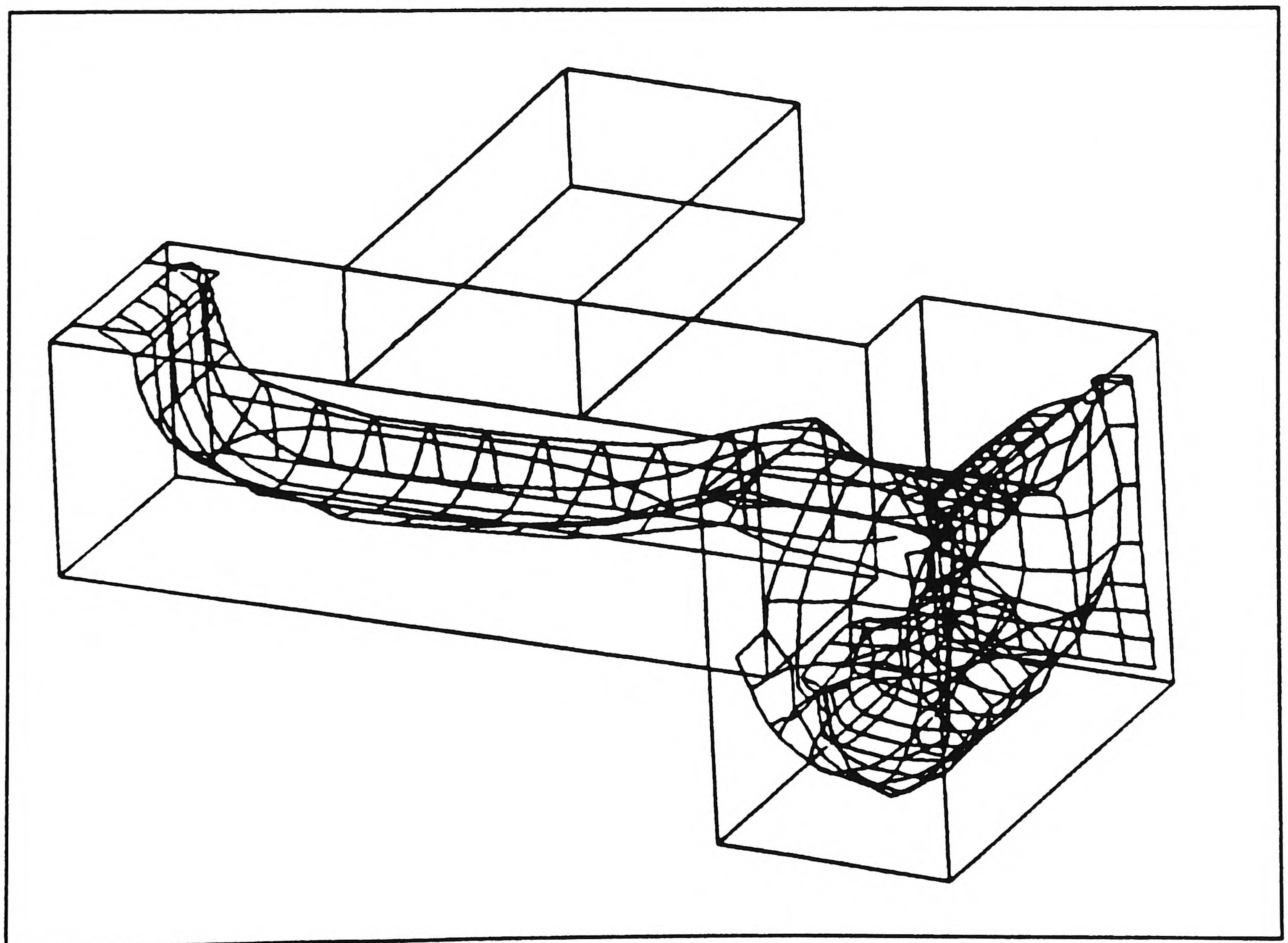


Figure 4.25 Liquid surface after 0.21 second.

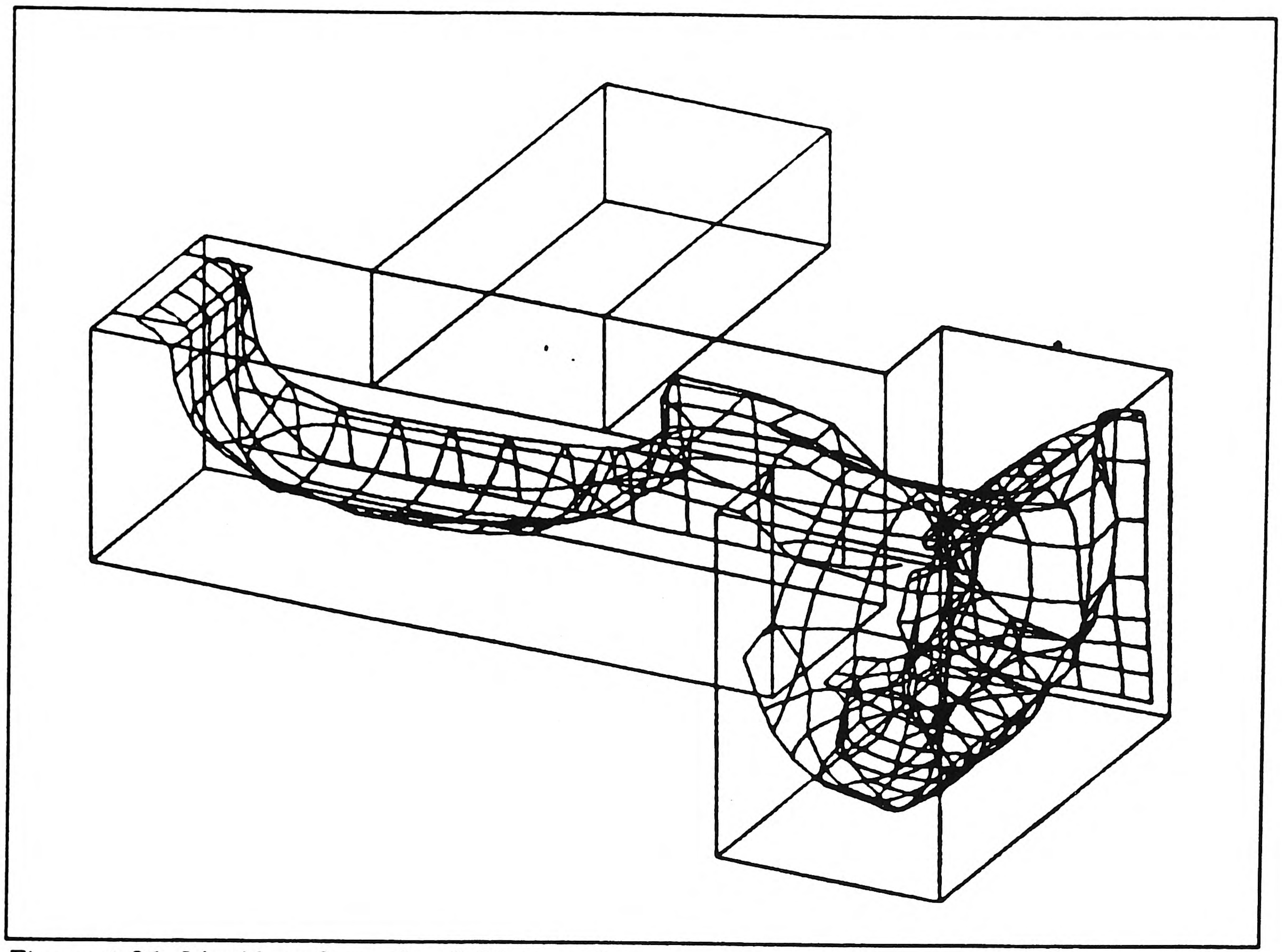


Figure 4.26 Liquid surface after 0.24 second.

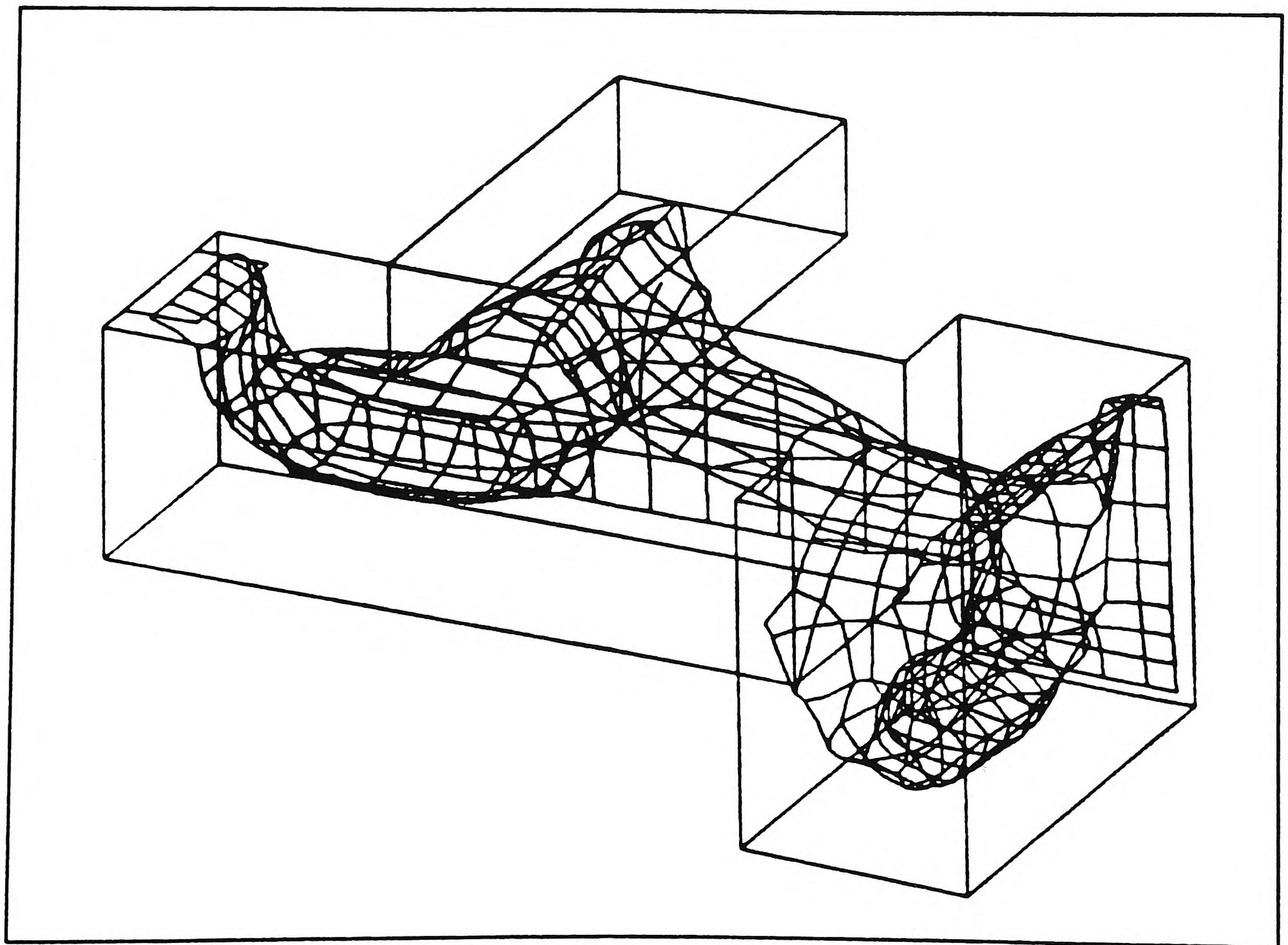


Figure 4.27 Liquid surface after 0.3 second.

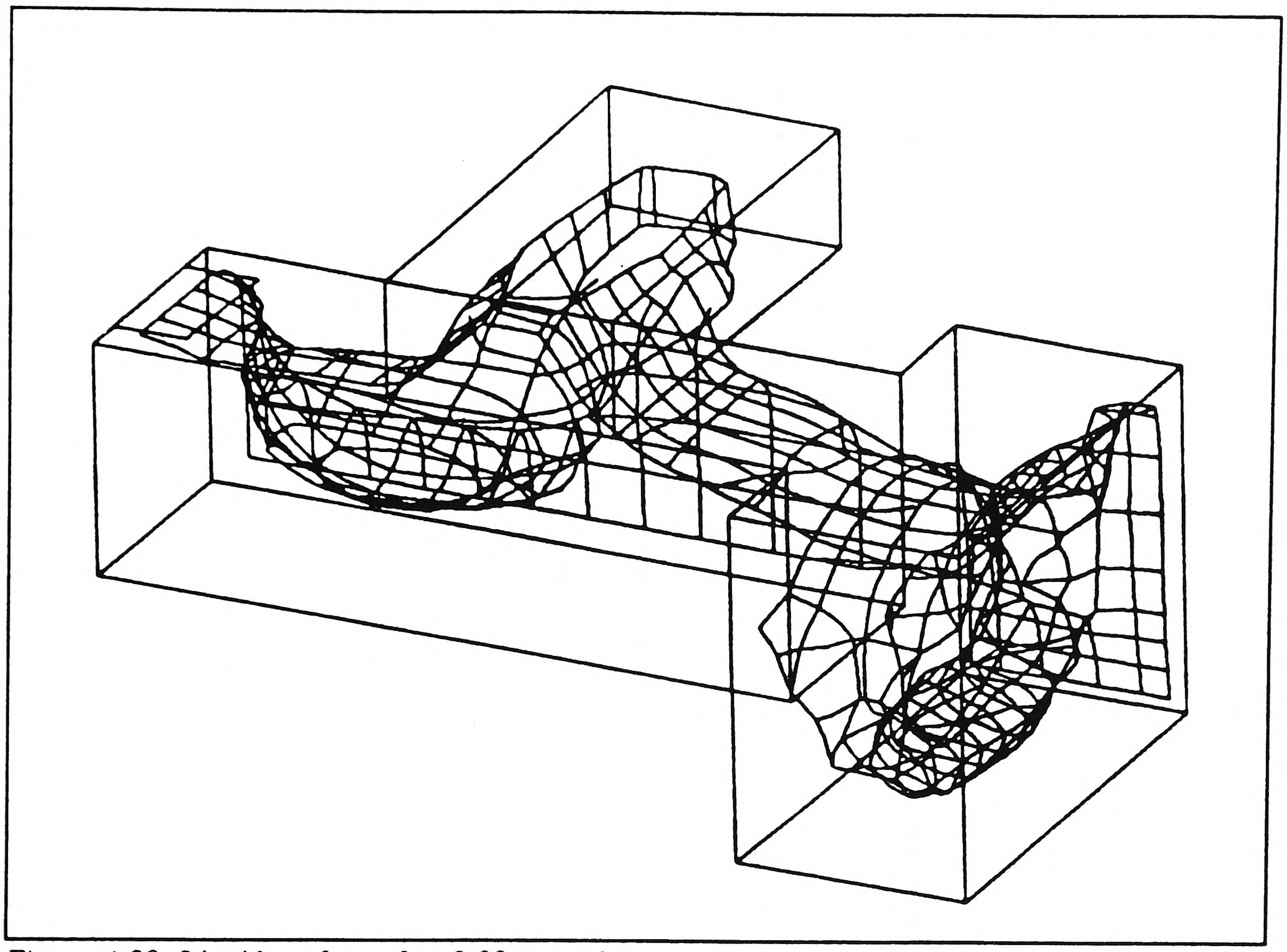


Figure 4.28 Liquid surface after 0.33 second.

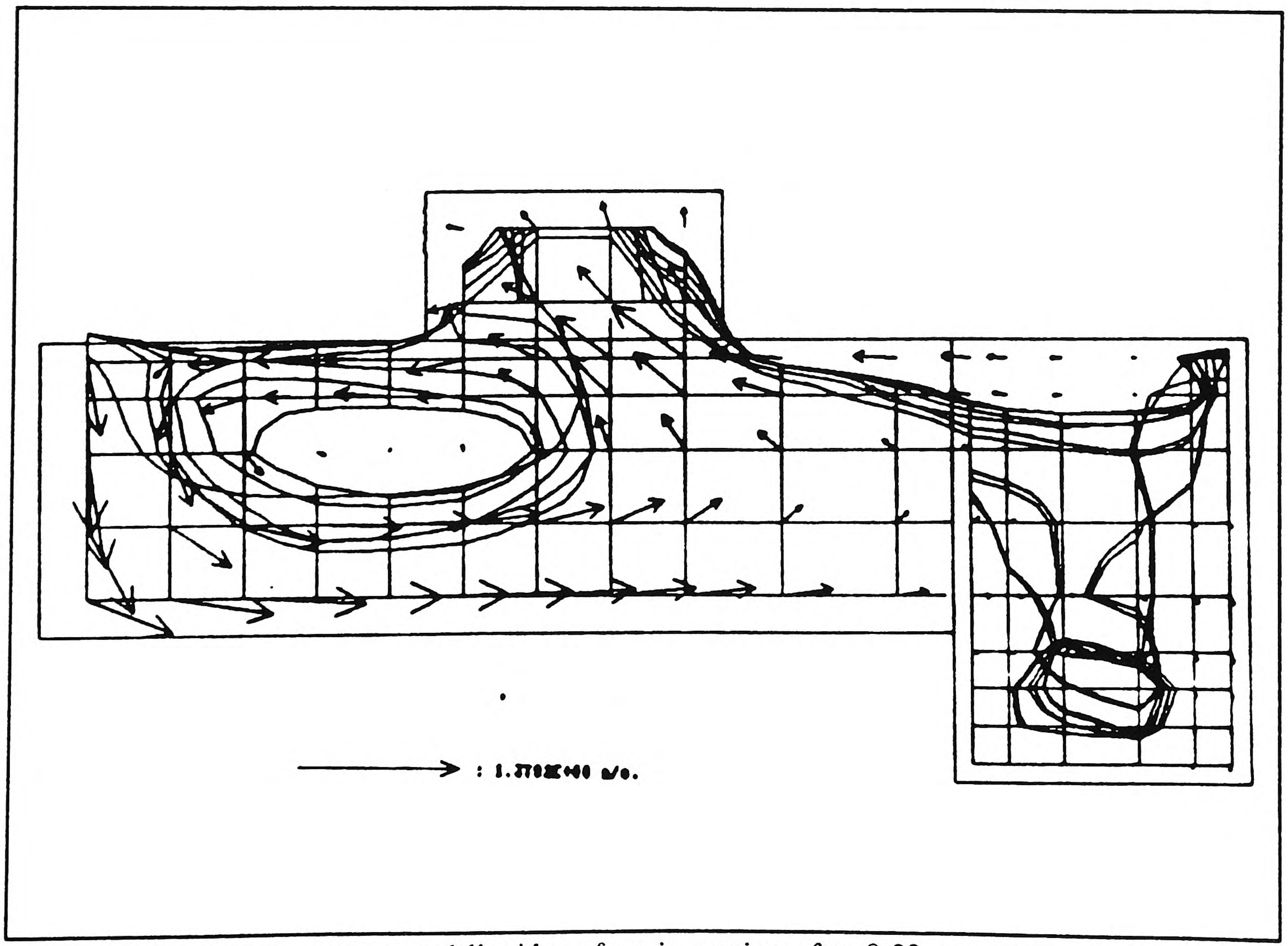


Figure 4.29 Velocity vectors and liquid surface in section after 0.33 second.

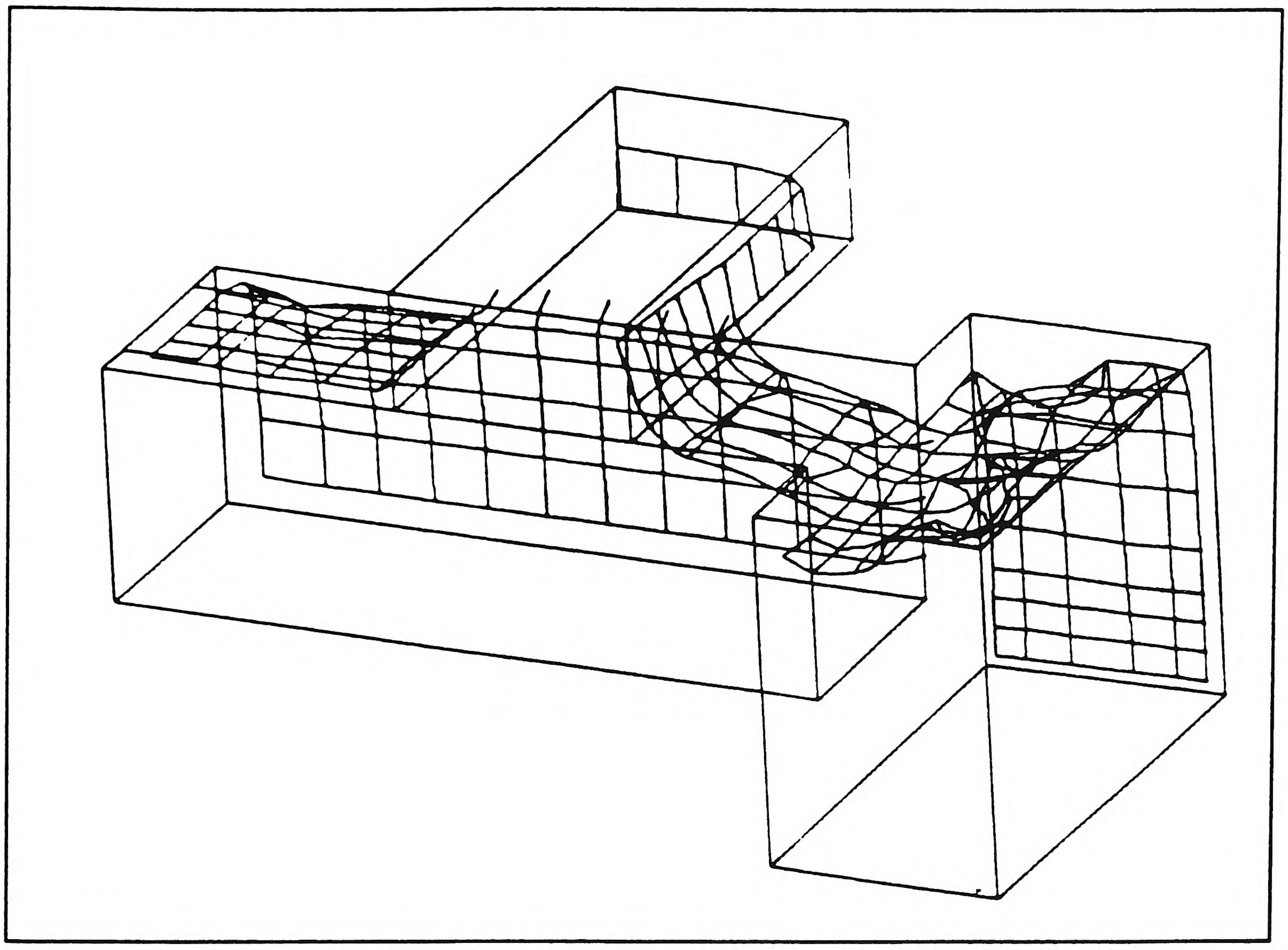


Figure 4.30 Liquid surface after 0.75 second.

4.4.4 A Proposed Alternative Geometry for the Gating System

In order to alleviate the wave problem and reduce the possibility of air being trapped in the casting, the author suggested to modify the original geometry by placing the ingate directly on top of the dross trap. See *Figure 4.31* for the proposed geometry. A coarser grid of size 10 x 8 x 15 = 1200 cells was used to test this geometry. The same time step size, boundary conditions and other run time parameters as described in *Industrial Case I* above, except the number of total time steps for the run is reduced to approximately 400.

4.4.5 Results

Figure 4.31 (at 0.12 second) shows how the jet of liquid is about to impact upon the end wall of the dross trap. *Figures 4.32* to *4.33* show the flow after impact and the subsequent behaviour of the liquid surface which is similar to those described before. However, later time steps indicate that less air is trapped (*Figure 4.33*, after 0.36 second), the post-impact is absent, and the entire running system is almost full before any liquid leaves through the outlet.

4.4.6 Remarks for the Proposed Geometry Case

The advantage of having computer simulation software to aid the design and production of castings has been illustrated. With respect to the foundry industry, some of the aspects highlighted were:

1. Relatively easy to modify the geometry of the running system to investigate its effects on the fluid dynamics of the filling metals.

2. Considerable savings in time, labour and material costs can be achieved at the design and manufacturing stages of producing the mould patterns of the castings. The mould patterns are required when setting up the sand moulds for the casting. A mould pattern is usually made of wood. It must have the correct shape and slightly enlarged dimensions (to account for metal contraction) of the casting to be cast.

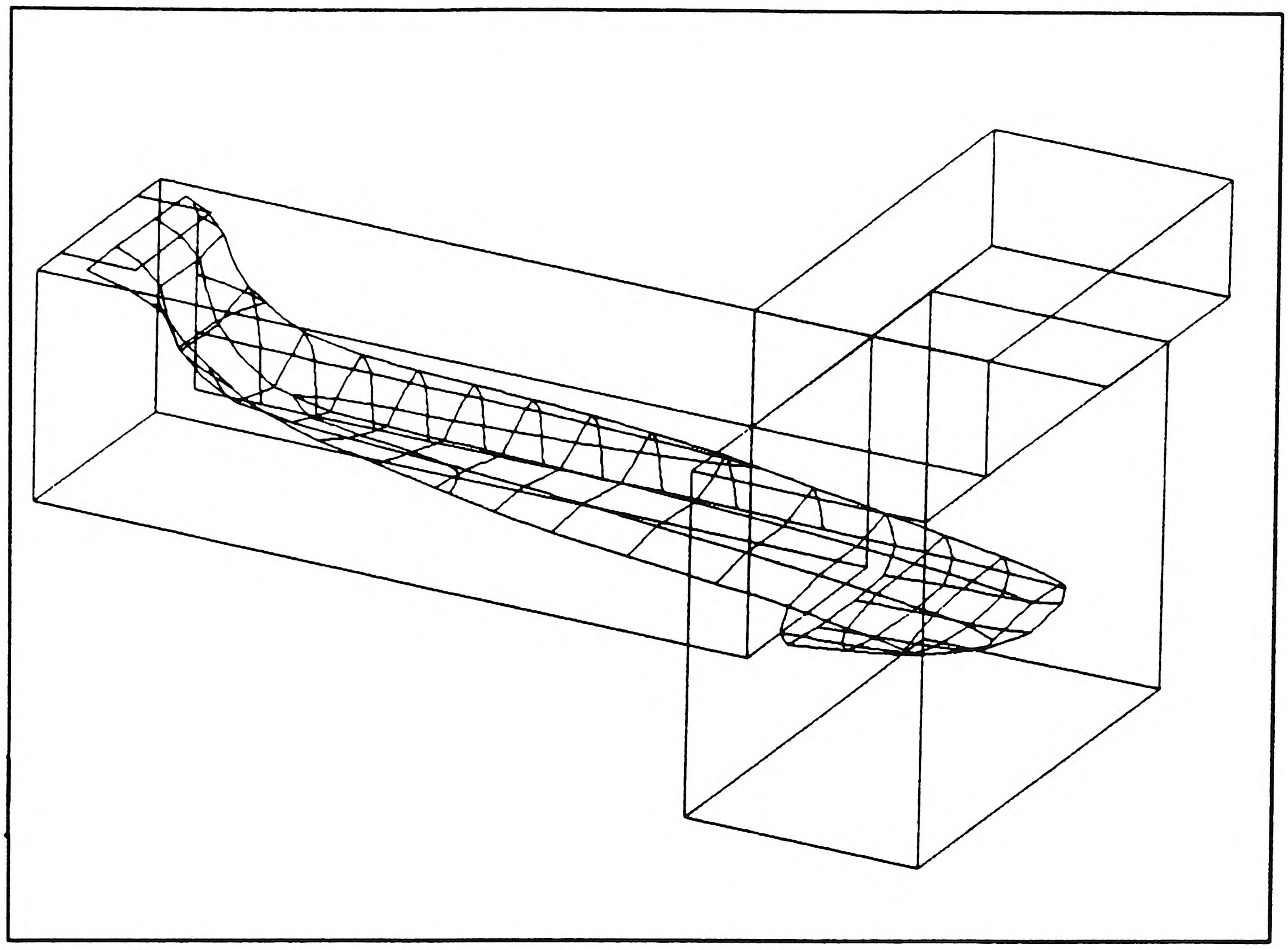


Figure 4.31 Alternative geometry: Liquid surface after 0.12 second.

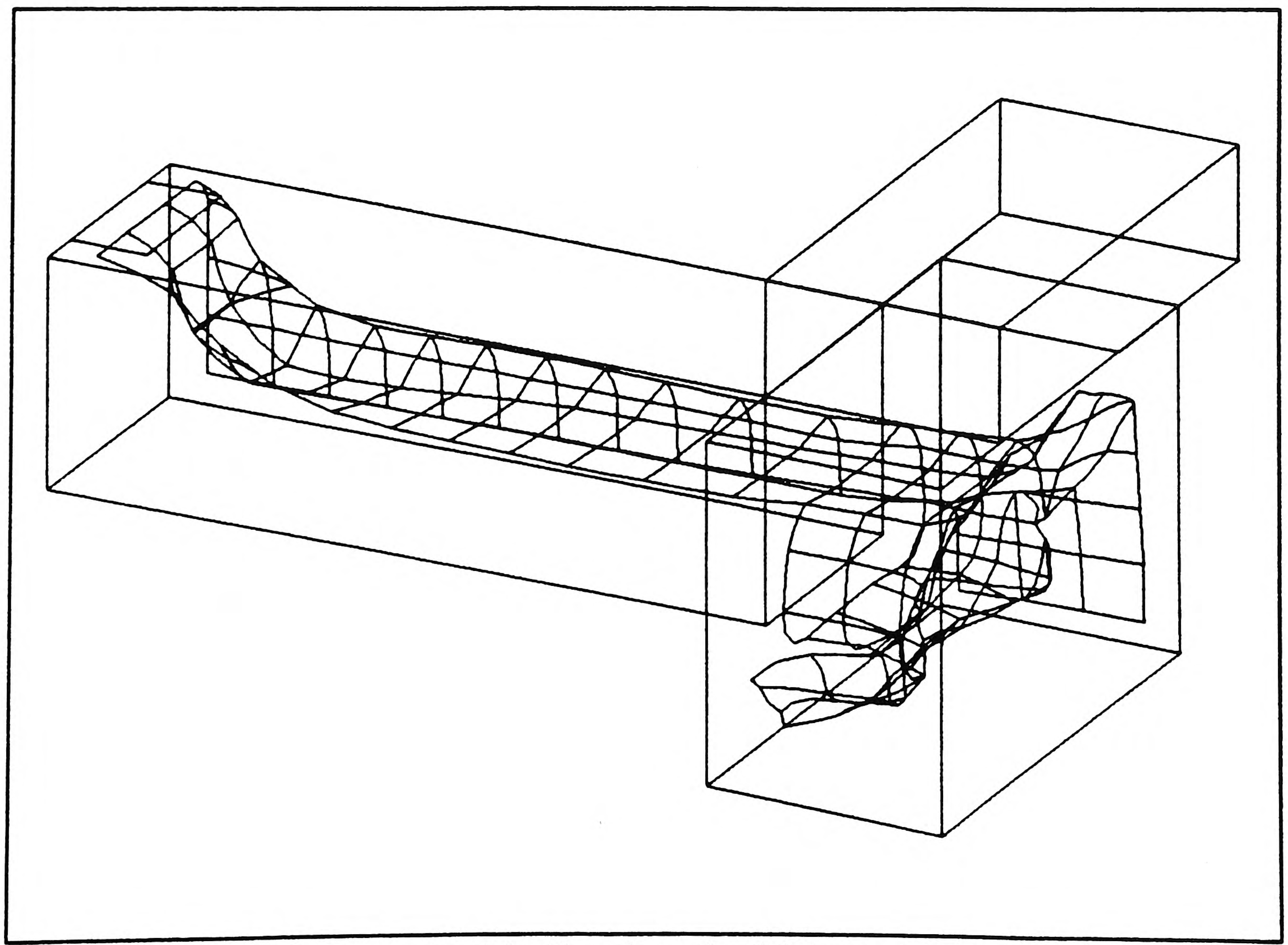


Figure 4.32 Alternative geometry: Liquid surface after 0.24 second.

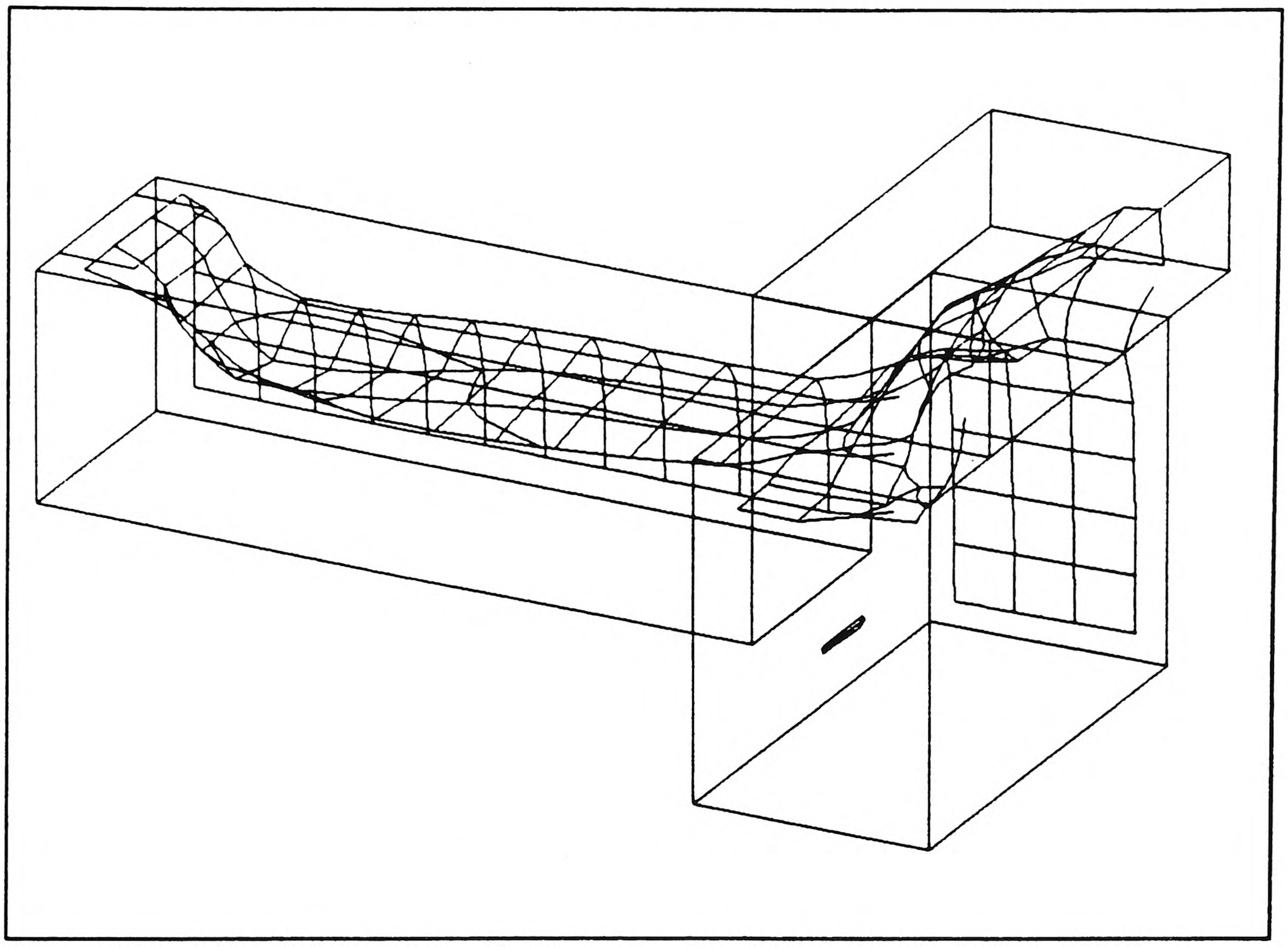


Figure 4.33 Alternative geometry: Liquid surface after 0.36 second.

4.5 Industrial Case II: Step Wedge

An aluminium alloy casting in the shape of a step wedge (*Figures 4.34a* and *4.34b* which are not drawn to scale) was cast in a sand mould at a local foundry in 1992 [Preddy (1992)]. The purpose of producing such a casting was to test the effect of the running system on its quality.

4.5.1 Problem Specification and Computational Details

The geometry and dimensions for the step wedge is shown in *Figures 4.34a* and *4.34b* respectively. The mesh used to model the step wedge in three-dimensions is depicted in *Figure 4.35.* Similarly, the mesh for the two-dimensional case is shown in *Figure 4.36.* A three-dimensional simulation consisting of 20 x 23 x 5 = 2300 cells and a grid of size 20 x 28 = 560 cells in two-dimensions were used to carry out the simulations. The notion of using both of a two- and three-dimensional versions of SEA filling algorithm was to investigate the effect of the third dimension in this almost axially-symmetric example; see *Figure 4.34a.* The result is quite surprising as shown in *Section 4.5.2.*

The material properties and initial conditions used in both runs are shown in *Table 4.11* and *Table 4.12.*

	Aluminium Alloy	Air
Density, ρ (Kg/m^3)	2670.0	1.0
Kinematic Viscosity, ν (m^2/s)	2.0x10^{-5}	10^{-5}

Table 4.11 Material properties for industrial case II.

The run time parameters for the two- and three-dimensional cases are shown in *Tables 4.13* and *4.14* respectively.

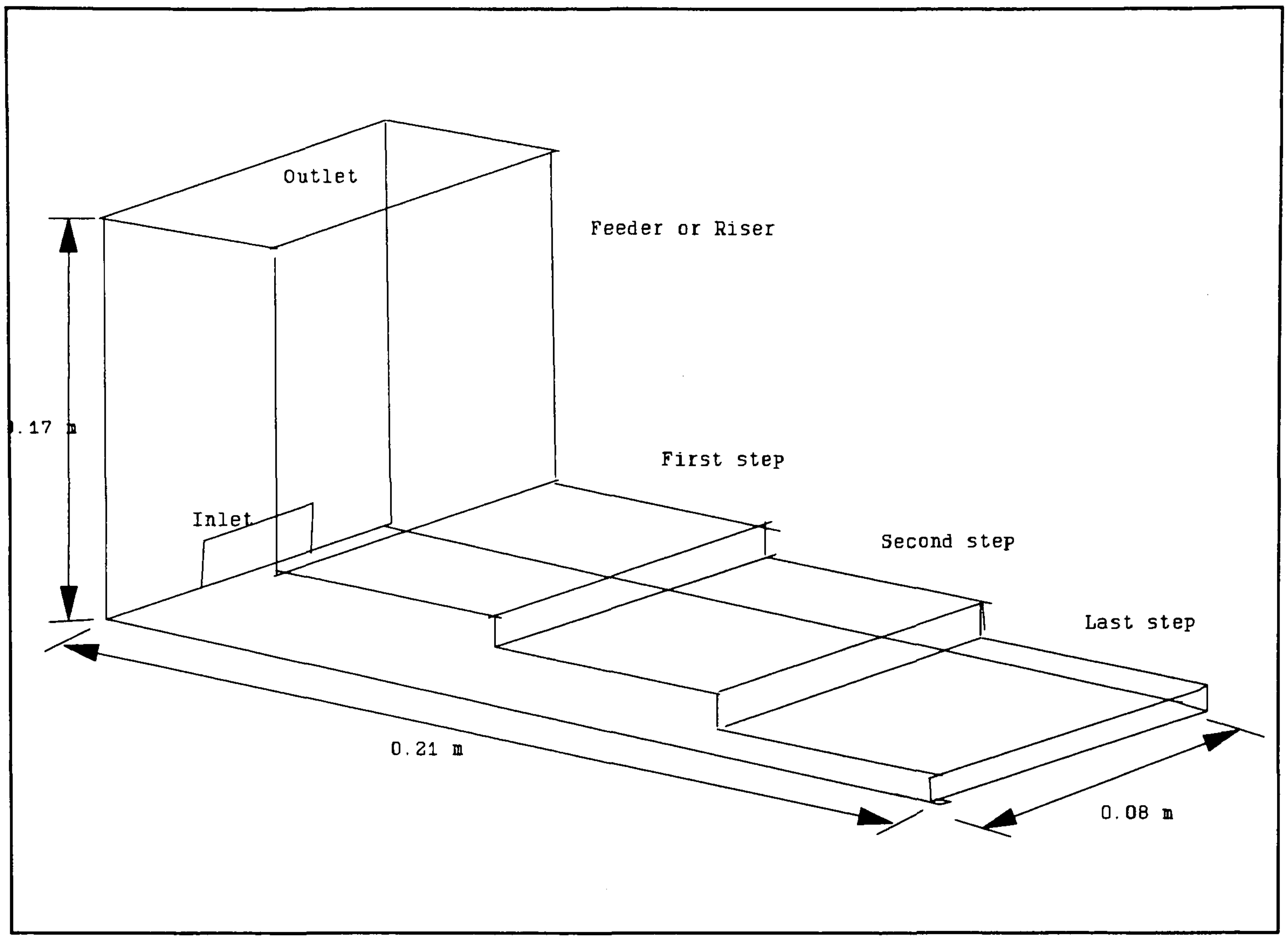


Figure 4.34a Geometry of the step wedge in 3-D.

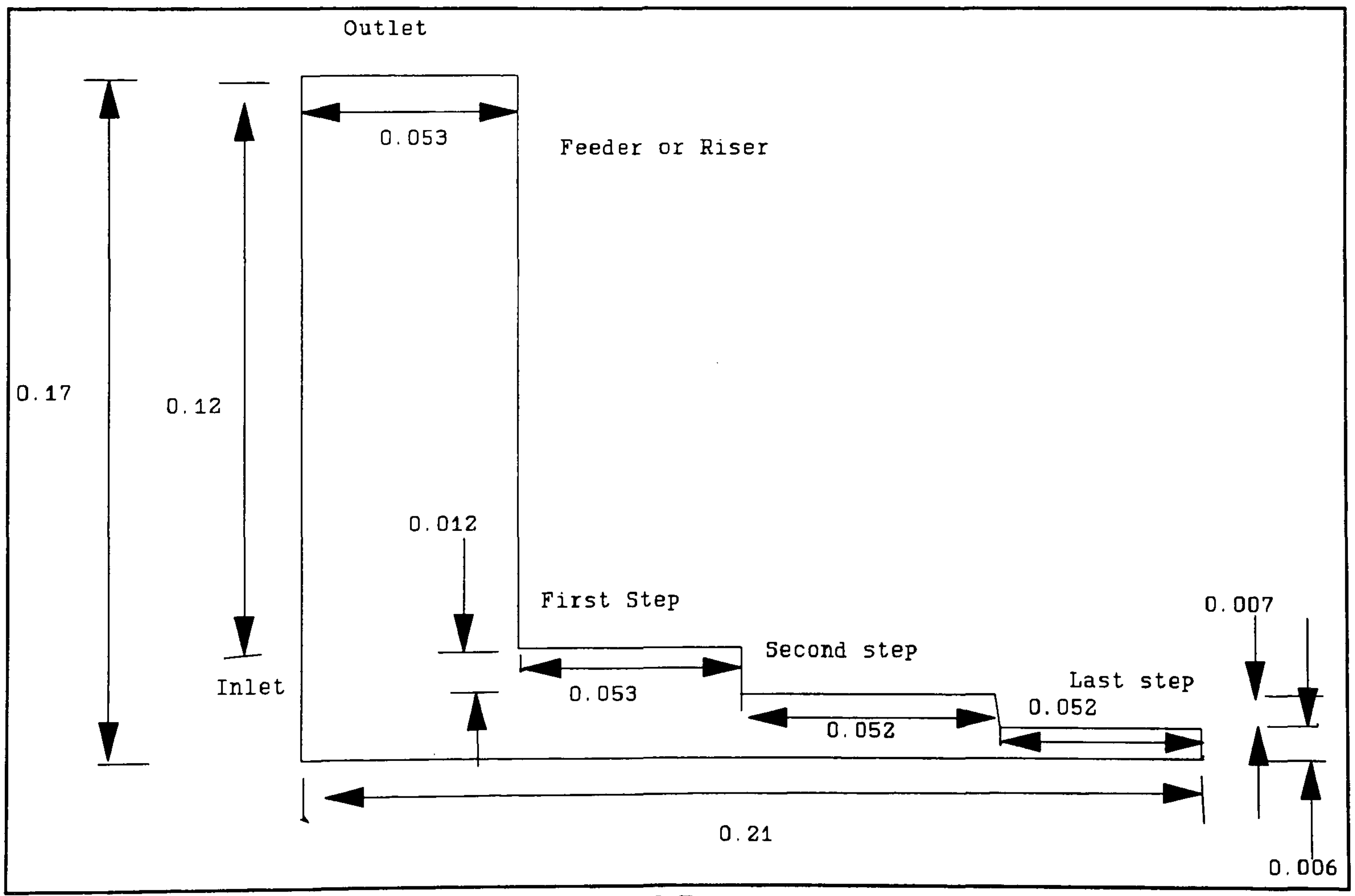


Figure 4.34b Geometry of the step wedge in 2-D.

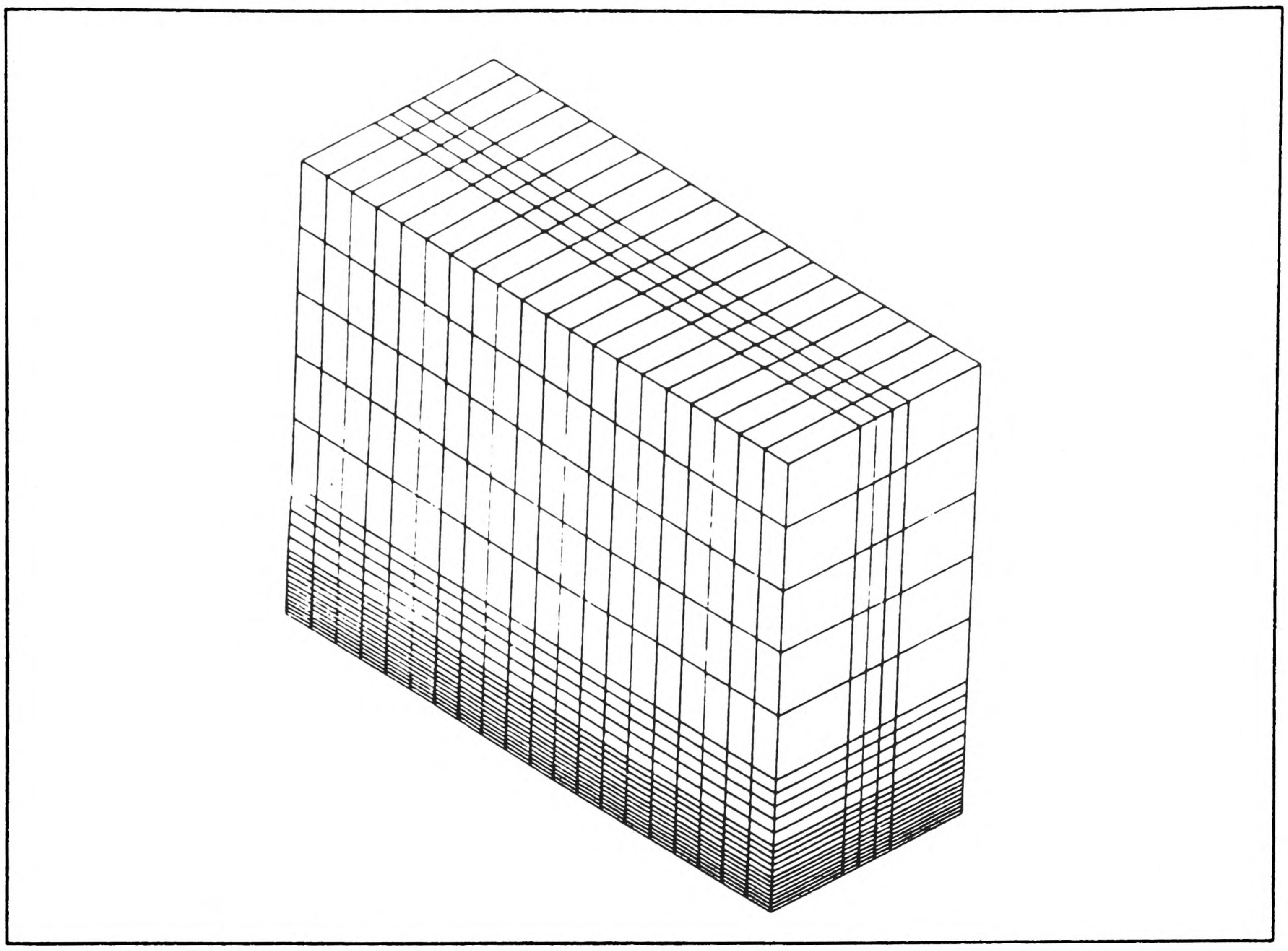


Figure 4.35 The mesh used to model the step wedge in 3-D.

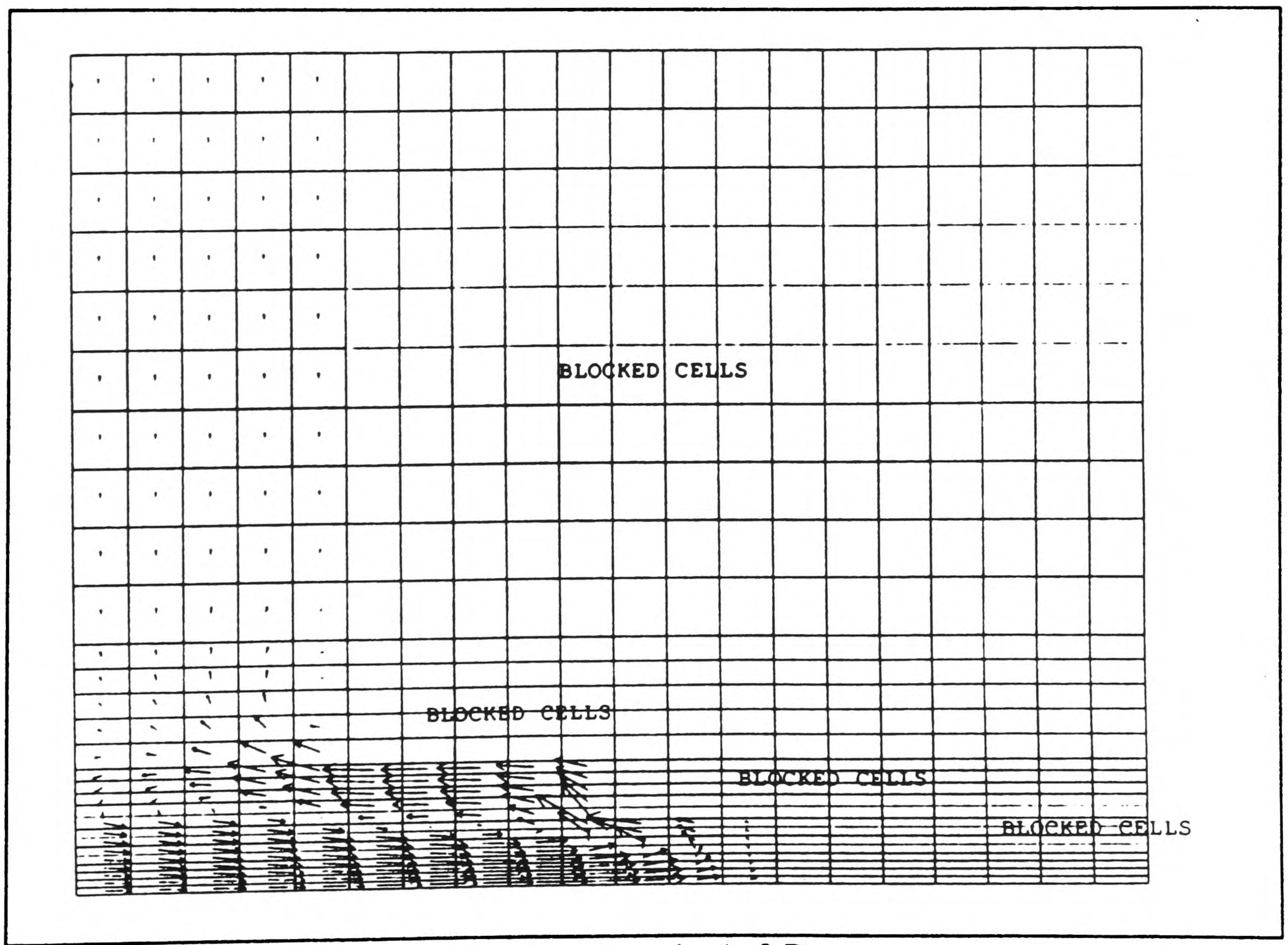


Figure 4.36 The mesh used to model the step wedge in 2-D.

Initial conditions:

Inflow velocity : 0.25 m/s

Outlet (along the top of the feeder): Pressure at outlet boundary is prescribed to zero.

Boundary conditions:

Walls: impermeable, isothermal, and non-slip or zero velocity at walls is assumed.

Gravity acting downward: 9.81 ms^{-2}

Other information:

For the 3-D case:	Liquid volume flow rate : $6.5x10^{-5}$ m^3/s
	Volume of mould cavity: $8x10^{-4}$ m^3
	Expected fill-up time : 12.3 s
For the 2-D case:	Liquid volume flow rate : $3.25x10^{-3}$ m^3/s
	Volume of mould cavity: $9.998x10^{-3}$ m^3
	Expected fill-up time : 3.07 s

Table 4.12 Initial and boundary conditions for industrial case II.

Relaxation factors:

Linear: pressure (0.8)

False time-step: u(0.03), v(0.03)

Time-step size: 0.0025 s

Residual tolerance for velocities and pressure: 10^{-6}

Number of sweep per time-step: 50

Total number of time steps: 500

Simulated real time: 1.25 seconds

Total run-time (wall clock): approximately 2 hours (Sun Sparc IPX)

Table 4.13 Run-time parameters for the 2-D step wedge case

Relaxation factors:

Linear: pressure (0.8)

False time-step: u(0.03), v(0.03), w(0.03)

Time-step size: 0.005 s

Residual tolerance for velocities and pressure: 10^{-6}

Number of sweep per time step: 50

Total number of time-steps: 2680

Simulated real time: 13.4 seconds

Total run-time (wall clock): approximately 10 hours (Sun Sparc IPX)

Table 4.14 Run-time parameters for the 3-D step wedge case.

4.5.2 Results

• Two-dimensional Case:

After 0.1 second (*Figure 4.37*), liquid metal flows into the mould with an initial velocity of 0.25 m/s. The front of the liquid nearly reaches the end of the first step. Strong air movement is recorded just above the liquid front around the first step.

Almost the entire length of the step is filled with liquid metal after just 0.3 second, see *Figure 4.38*. Note that the air cavity at the end of the last step and a little air void in the upper left of the second step. Air is being forced out from the cavity along all the steps.

After 0.6 second (*Figure 4.39*), the liquid has filled most of the steps and reached the feeder itself. Since the mould is impermeable, the air cavity is trapped at the top part of the last and second step.

After 1.0 second (*Figure 4.40*), the mould is about 75% full. But some air still remains at the top part of the last step.

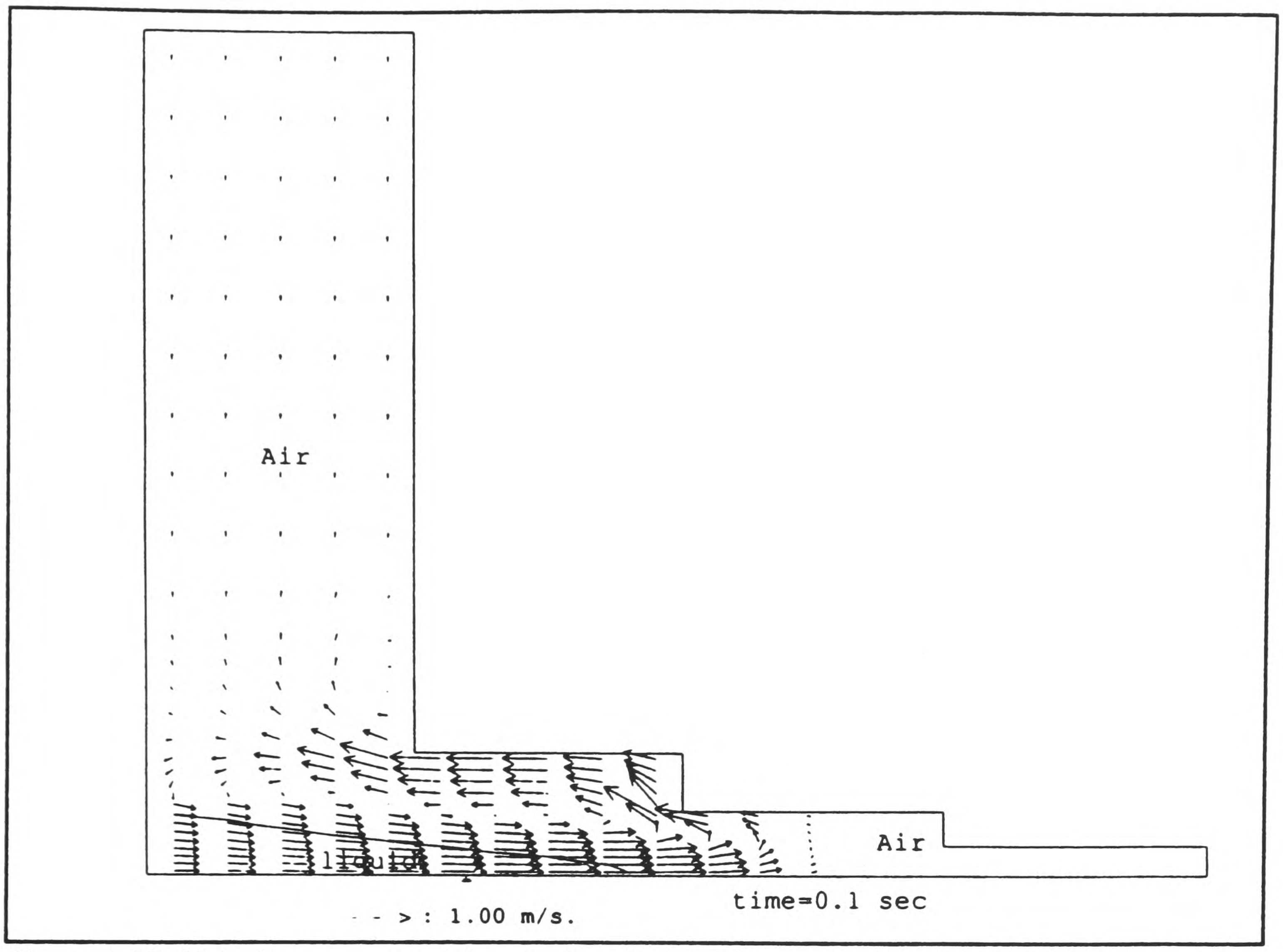


Figure 4.37 Velocity vectors and free surface in the 2-D step wedge after 0.1 second.

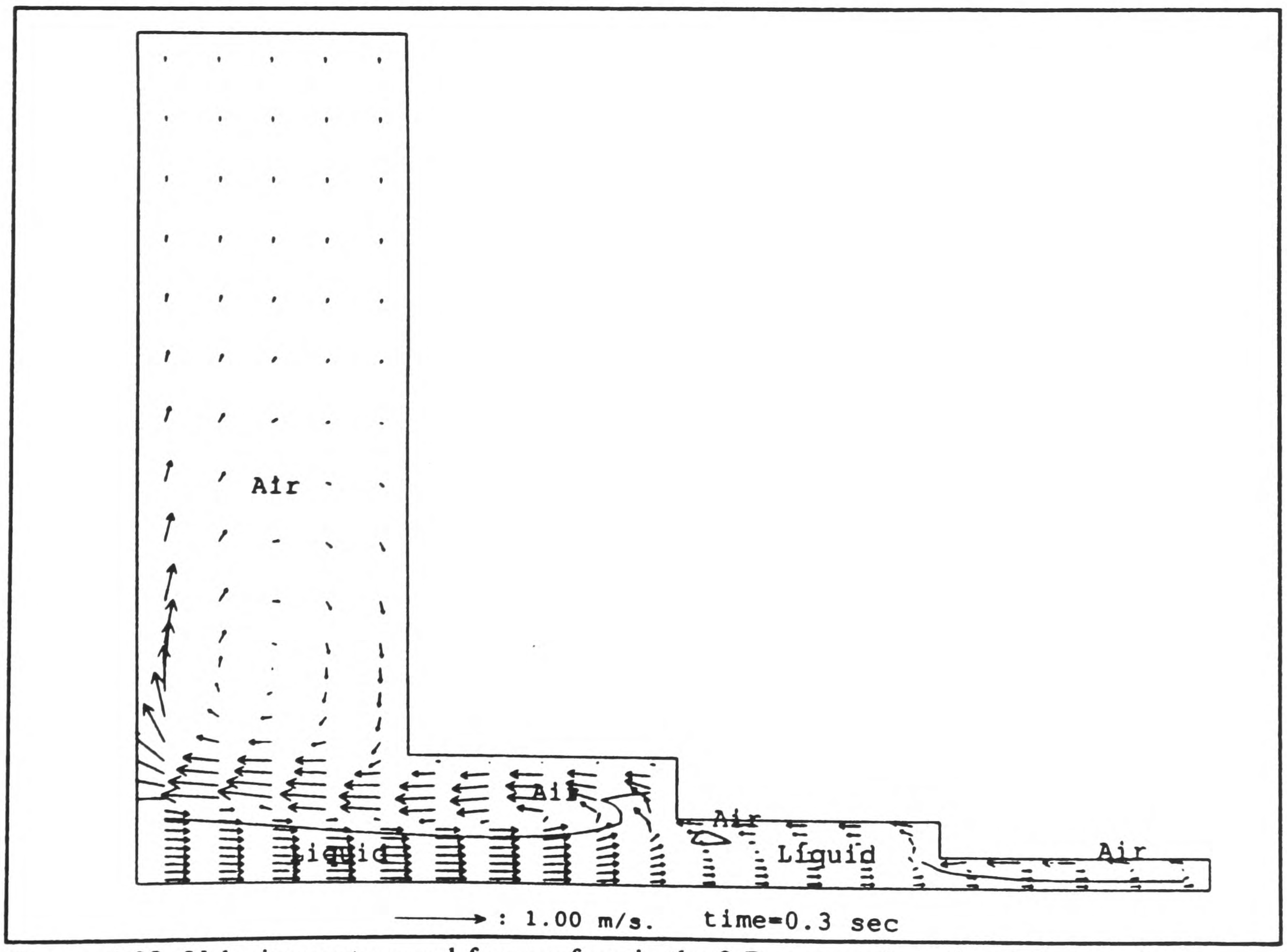


Figure 4.38 Velocity vectors and free surface in the 2-D step wedge after 0.3 second.

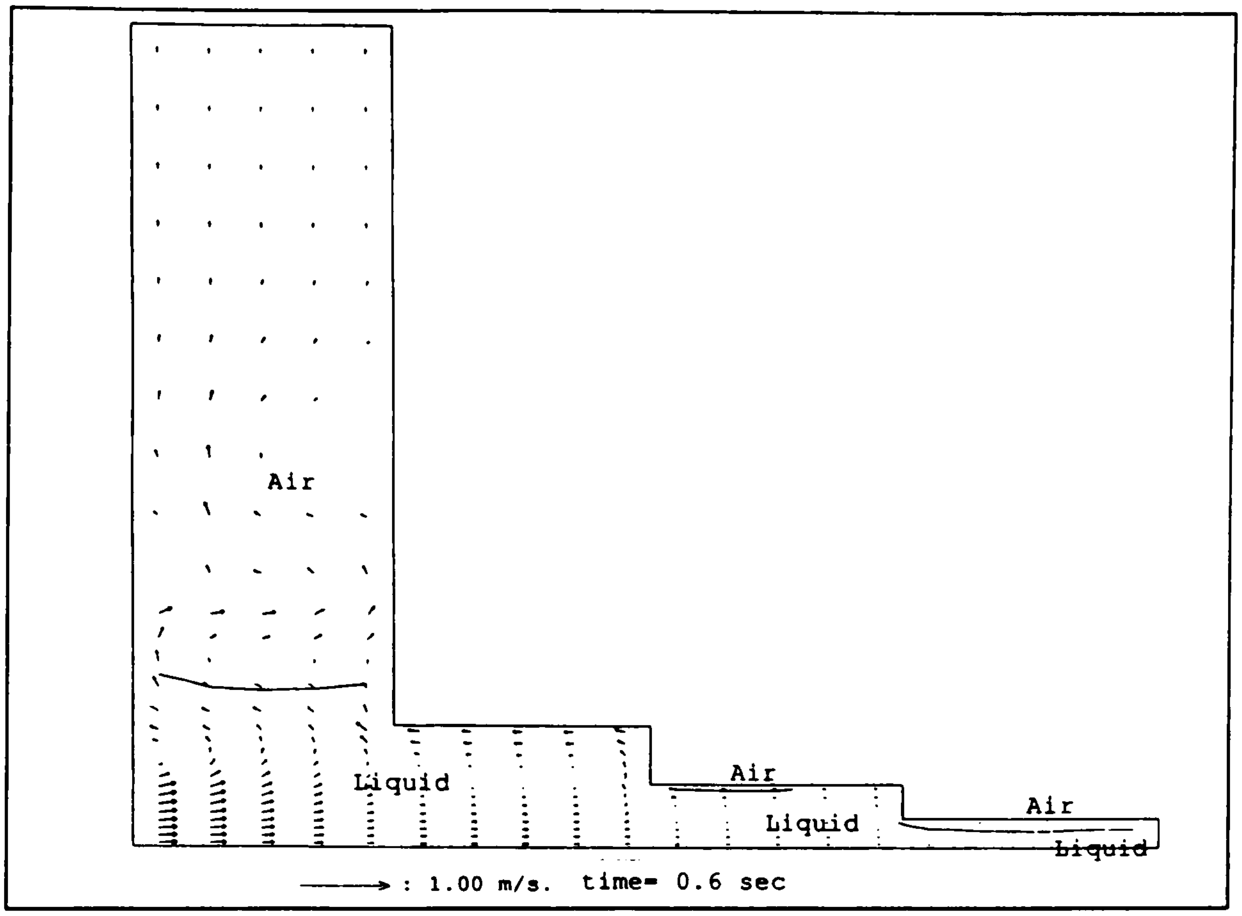


Figure 4.39 Velocity vectors and free surface in the 2-D step wedge after 0.6 second.

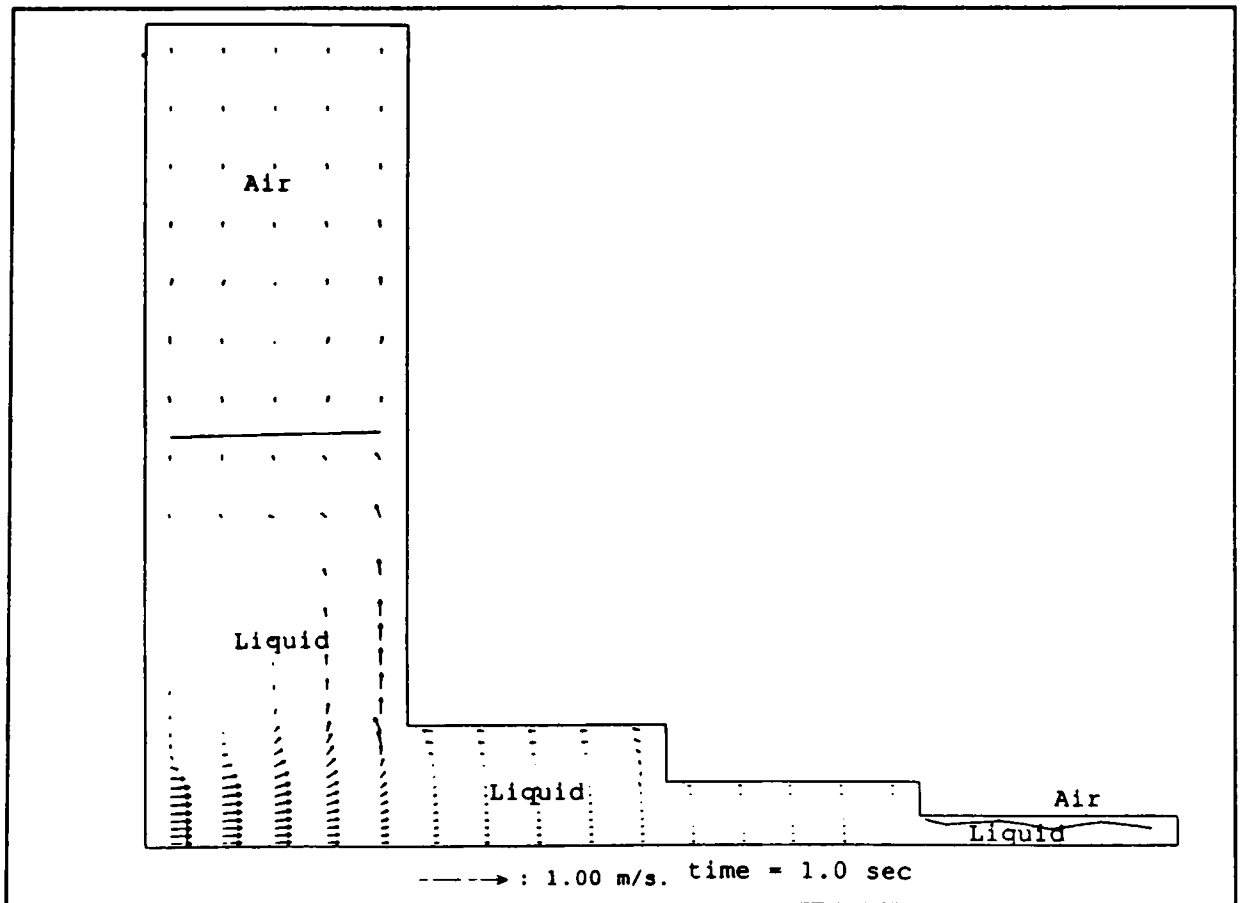


Figure 4.40 Velocity vectors and free surface in the 2-D step wedge after 1.0 second.

• **Three Dimensional Case:**

Figure 4.41 shows the fluid just entering the mould cavity via the inlet after 0.1 second. After 0.3 second (*Figure 4.42*), the fluid is spread out to the full width of the step and reaching the end of the first step. The shape of the fluid having a hub at the inlet and thinning out as it moves away from the inlet is reasonable according to K Preddy of Stone Foundry, Woolwich [Preddy (1993)]. The front of the fluid has advanced into the last step of the cast after 0.6 second as shown in *Figure 4.43*. After 4.0 seconds, the fluid has almost filled up all the steps except for some air pockets, as shown in *Figure 4.44*. After 13 seconds, the entire cast, including the feeder is almost filled as seen in *Figure 4.45*. Note that no air pockets are seen in *Figure 4.45*. This may be explained by the flow pattern of the fluid which is depicted by the vector plot. A recirculation of fluid appears on the right side of the step wedge which drives the air out to the atmosphere via the outlet at the top of the feeder.

4.5.3 Remarks for Industrial case II

The air trapped at the end of the last step (at the far end from the inlet) which showed up in two-dimensional simulation throughout all time steps had disappeared altogether in the three-dimensional simulation. The point to draw is that in order to predict flow behaviour realistically and accurately, modelling the problem in three dimensions is essential. This is particularly true when predicting the filling patterns of castings produced by foundries in real life.

In addition, a simulated real time of 13 seconds to fill the three-dimensional step wedge by the model is close to the actual time taken (which is about 11 seconds) to fill the mould at the foundry [Preddy (1993)]. The simulation took longer time to fill could due to the following factors:

1. The mould used in the model is not porous, hence air could not escape via the sand walls.
2. The inflow velocity of liquid metal at the ingate might not be constant during filling in practice.

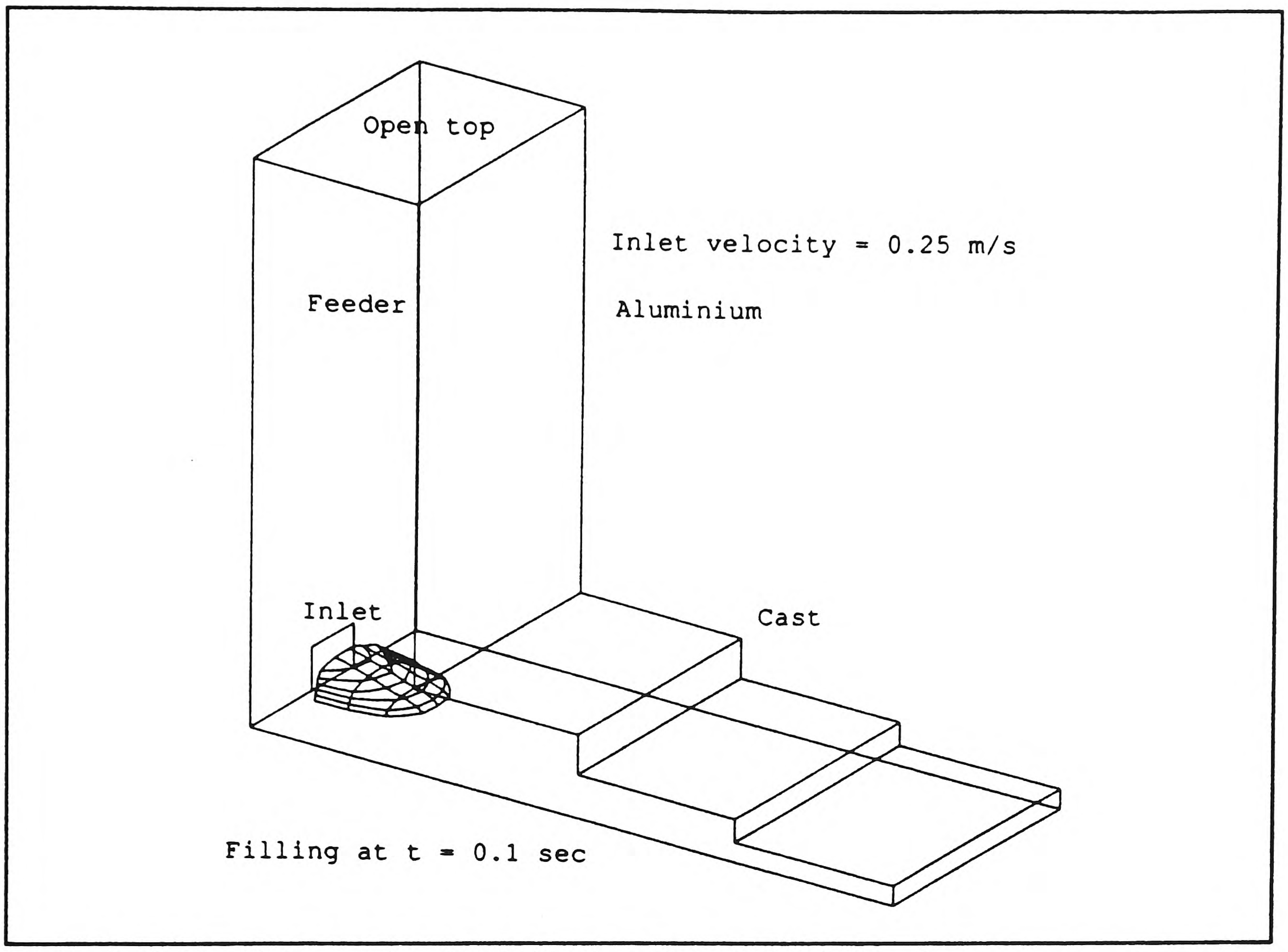


Figure 4.41 Liquid free surface in a 3-D step wedge after 0.1 second.

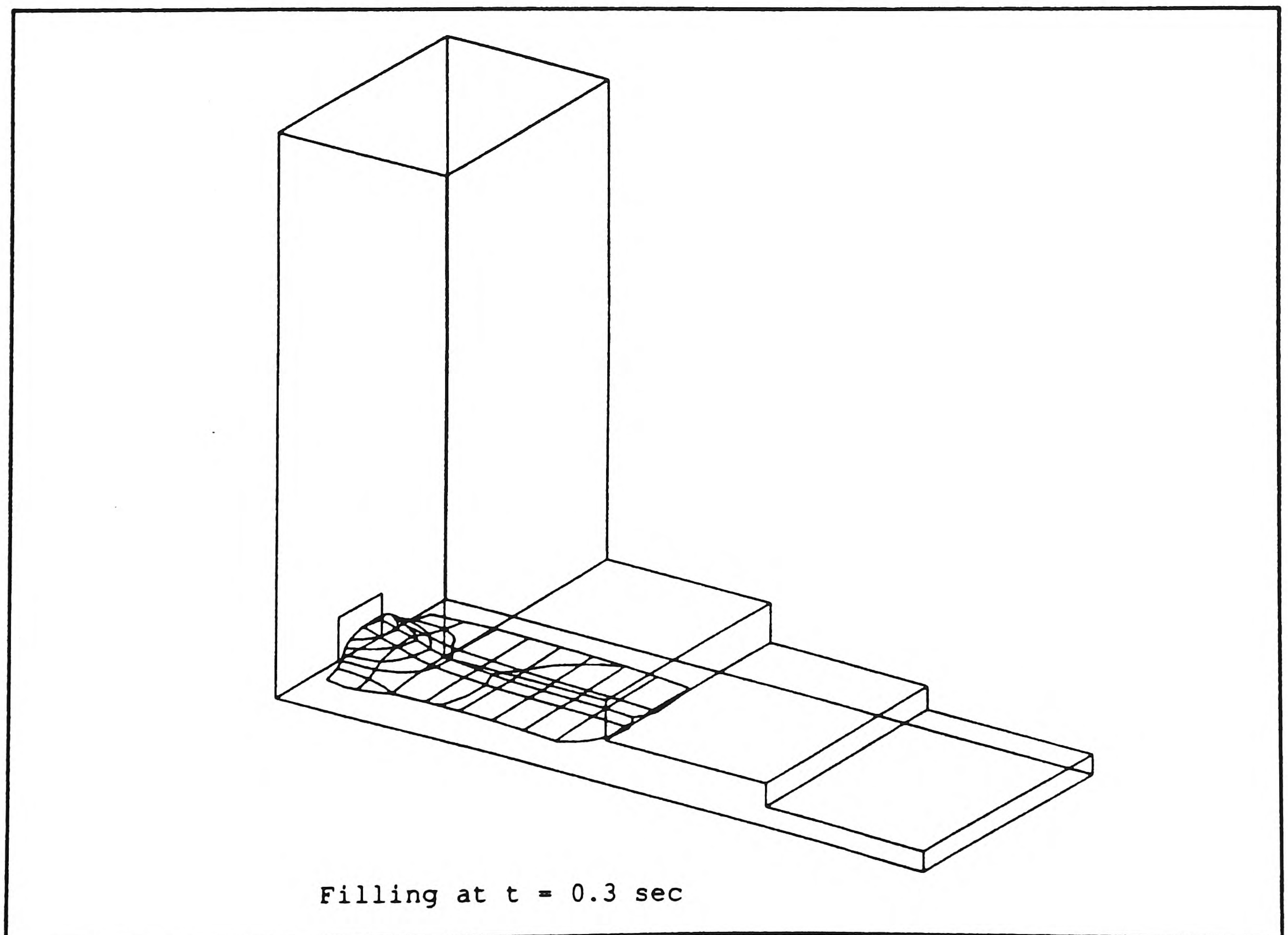


Figure 4.42 Liquid free surface at a 3-D step wedge after 0.3 second.

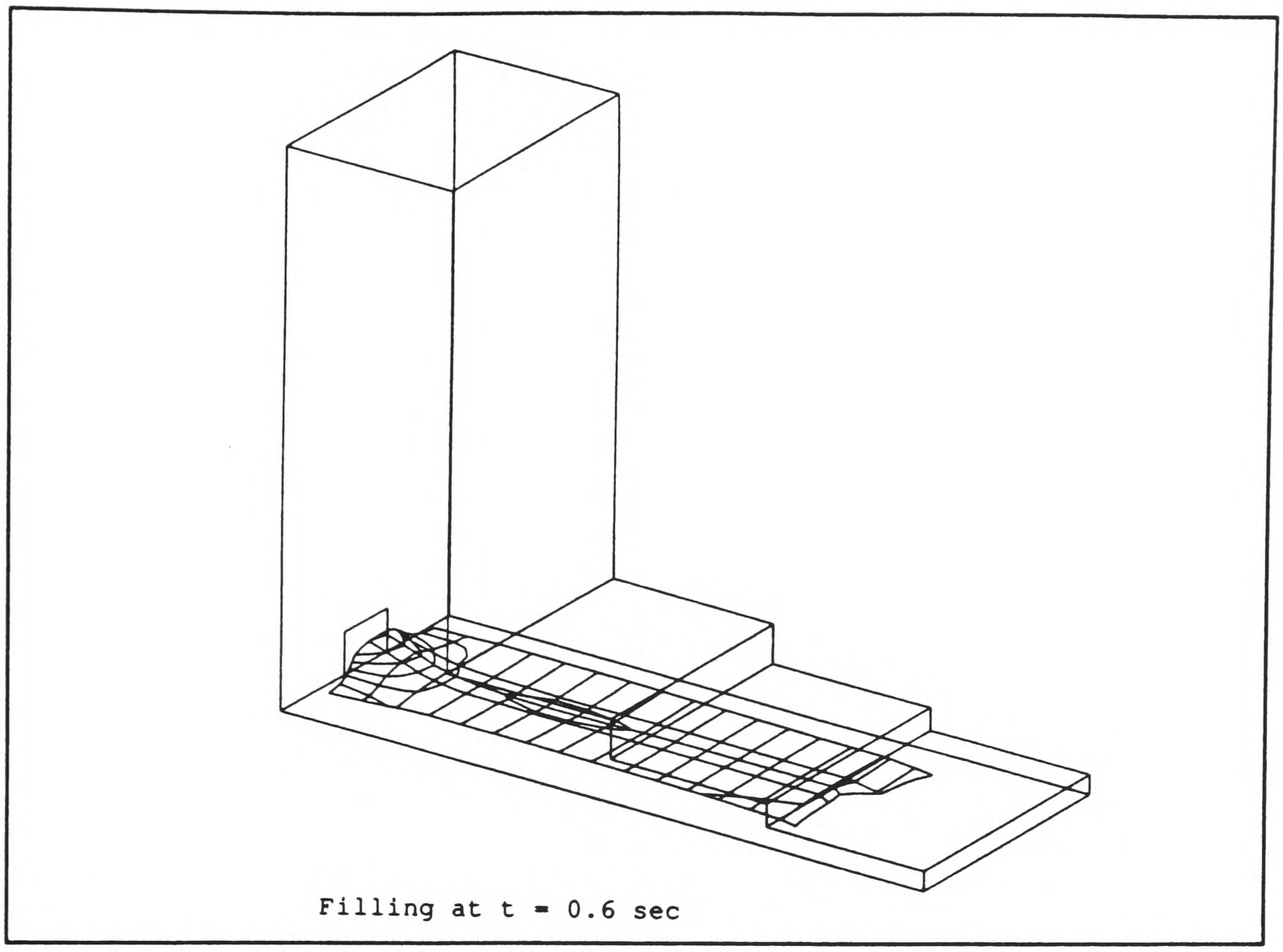


Figure 4.43 Liquid free surface in a 3-D step wedge after 0.6 second.

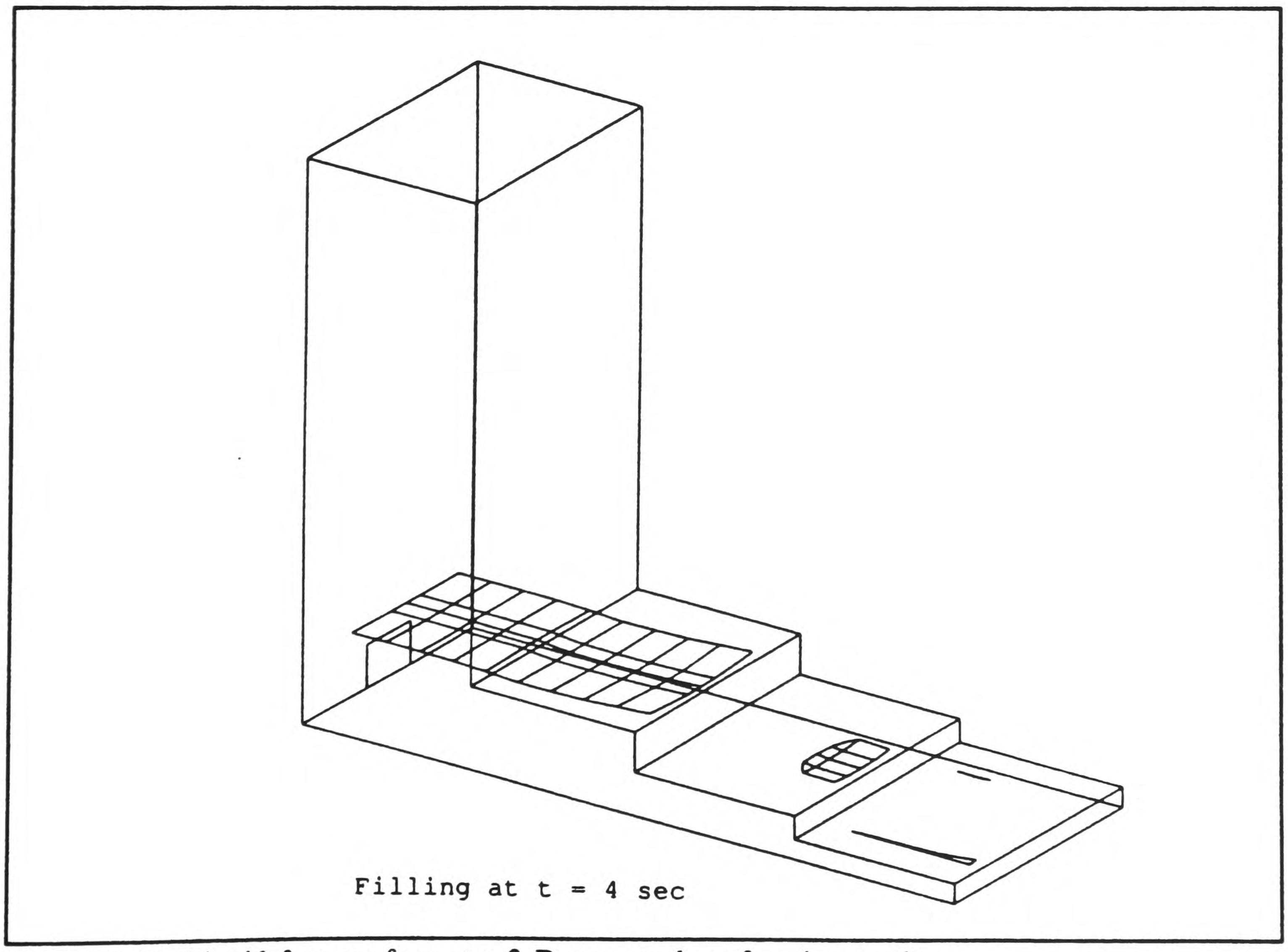


Figure 4.44 Liquid free surface at a 3-D step wedge after 4 seconds.

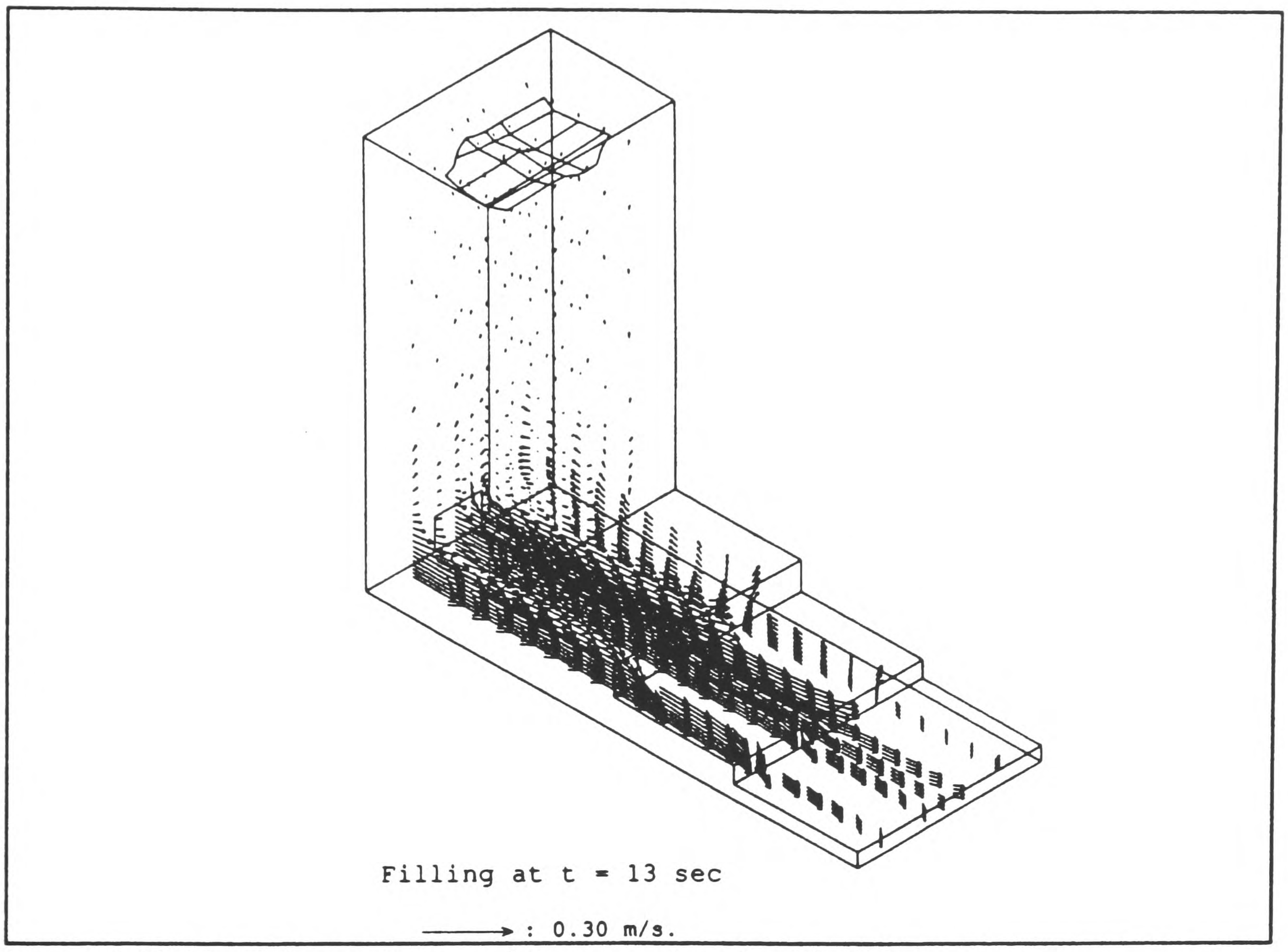


Figure 4.45 Liquid free surface at a 3-D step wedge after 13 seconds.

4.6 Academic Case: SEA Filling Algorithm vs SOLA-VOF and MAC methods

This case highlights the difference between the SEA and SOLA-VOF [Hirt and Nichols (1981)] and MAC [Harlow and Welch (1965)] methods. The geometry of the example for simulation is based upon that published by Lin and Hwang (1988) for the SOLA-VOF method, and that of Hwang and Stoehr (1987). No straight, quantitative comparison is to be made between Lin and Hwang's result and that of the author's, as the author used a slightly modified geometry. Furthermore, the author was primarily interested in the profile of the free surface produced by the three methods.

4.6.1 Problem Specification

The geometry used by Lin and Hwang is shown in *Figure 4.46* which depicts a vertical casting in three dimensions. The geometry used by Hwang and Stoehr (1987) is similar to that used by Lin and Hwang (1988) but smaller. The geometry adopted by the author is shown in *Figure 4.47* which is in fact the one used by Hwang and Stoehr (1987). The differences between Lin and Hwang's geometry and that used by the author are as follows:

1. The author's geometry is only in two-dimensions.
2. The ingate area in Lin and Hwang's geometry (*Figure 4.46*) is not included in the author's for the sake of simplicity.
3. The top edge of the mould cavity in the author's geometry is open, while in Lin and Hwang's geometry it is closed with a thick sand wall.

The reason why the modification is needed to Lin and Hwang's geometry (and also that of Hwang and Stoehr's) is that the SEA filling algorithm always solves for both air and liquid simultaneously. Therefore, it requires an outlet for the air to escape. From the conservation of volume point of view, the amount of liquid volume coming into the computational domain must equal the amount the volume of air leaving the domain, otherwise the solution will diverge.

On the other hand, the SOLA-VOF method does not solve for the air phase explicitly, even though air pressure within the domain could be estimated by the Bernoulli equation [Hwang and Stoehr (1987)].

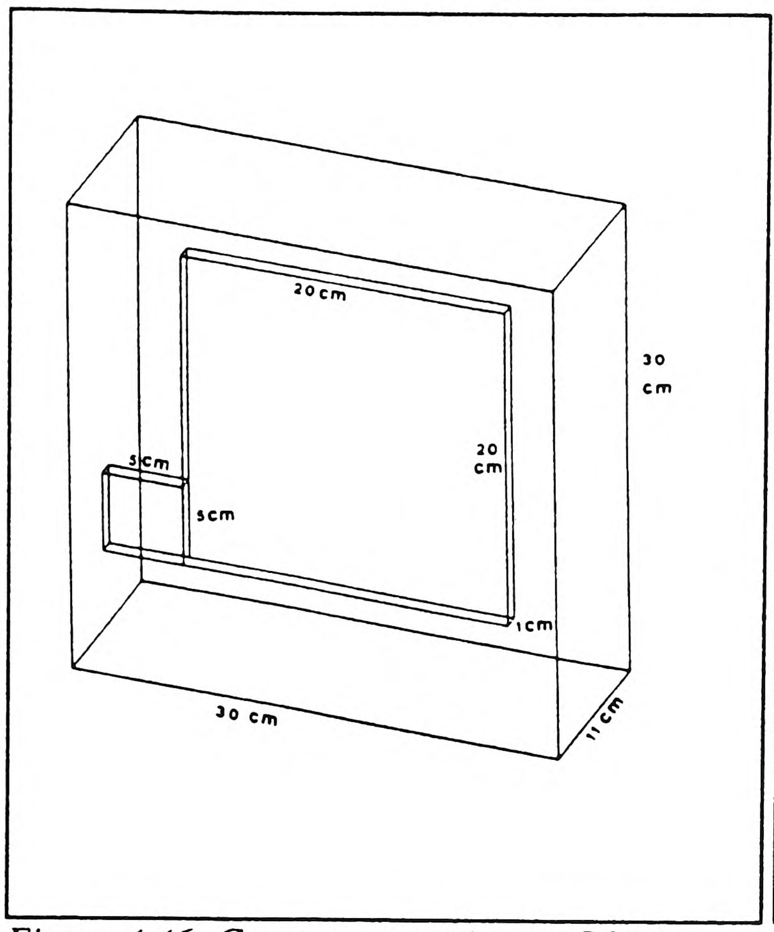


Figure 4.46 Geometry used by the SOLA-VOF case.

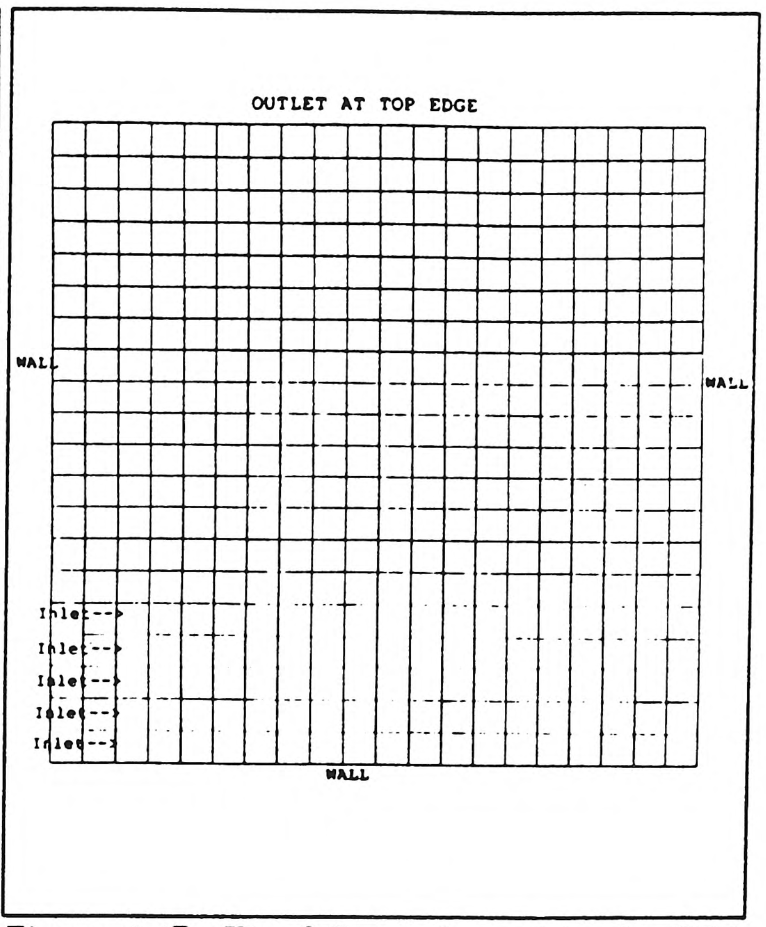


Figure 4.47 The 2-D mesh used in the SEA case.

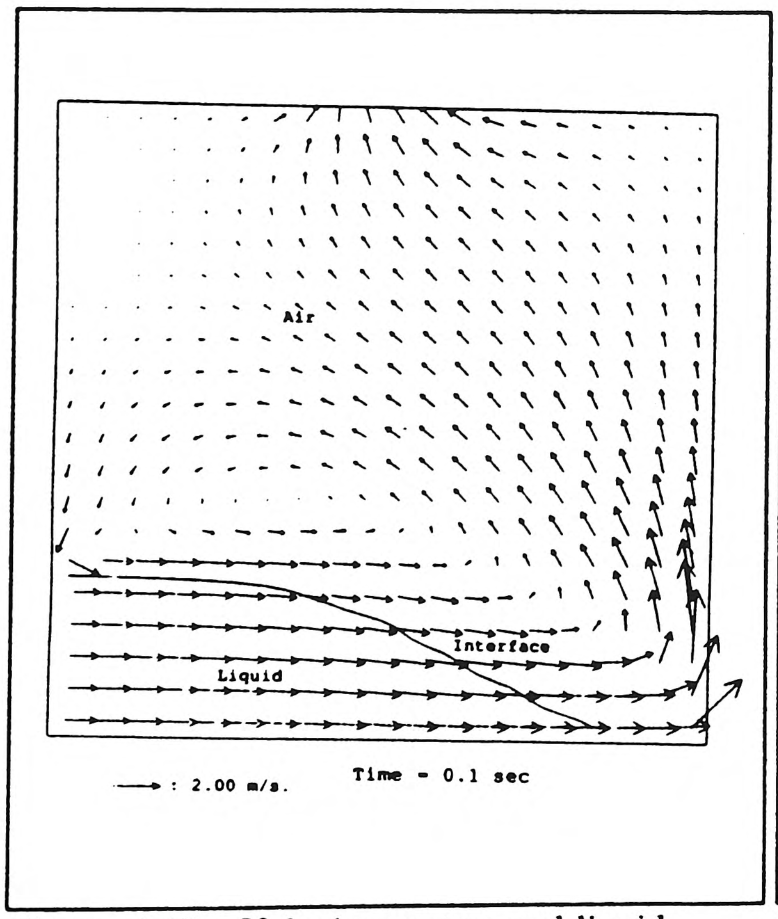


Figure 4.48a Velocity vectors and liquid surface after 0.1 second for the SEA case.

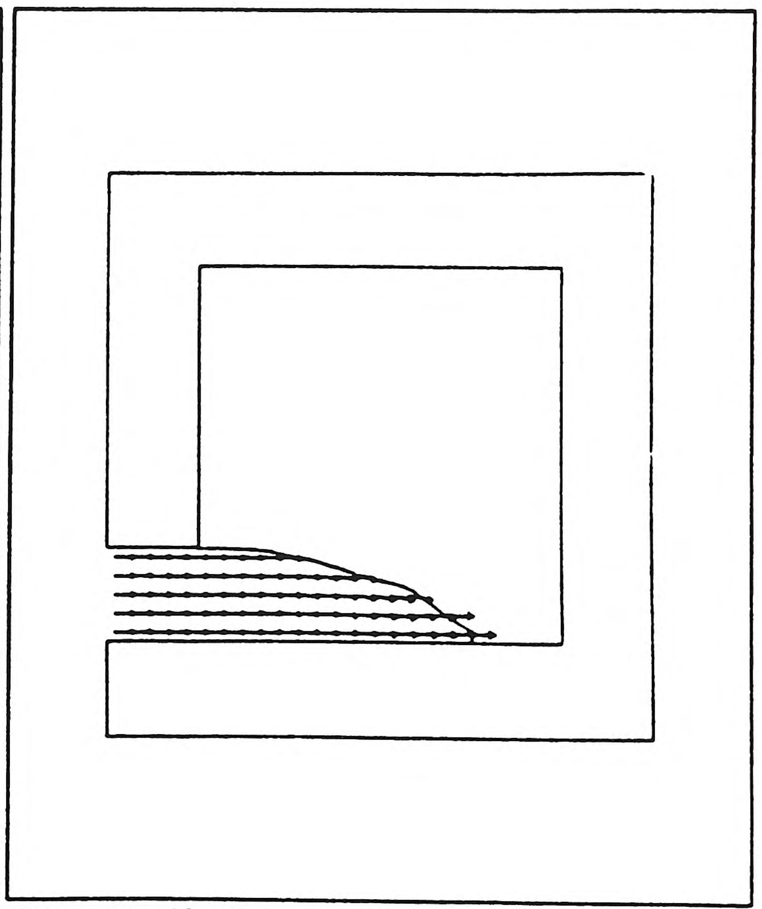


Figure 4.48b Velocity vectors and liquid surface after 0.1 second for the SOLA-VOF case.

4.6.2 Computational Details

A two-dimensional mesh of 20 x 20 = 400 cells was used to model this problem. The material properties and initial and boundary conditions are taken from Hwang and Stoehr's paper (1987). They are summarised in *Tables 4.15* and *4.16* below.

	Liquid	Air
Density, ρ (Kg/m^3)	8009.0	1.0
Kinematic Viscosity, ν (m^2/s)	$9.29\text{x}10^{-7}$	10^{-5}

Table 4.15 Material properties for the academic case.

The two-dimensional simulation was carried out using a Sun Sparc workstation and it took about 5 hours to complete. The run-time parameters are listed in *Table 4.17*.

4.6.3 Results

Figure 4.48a shows the liquid entering the mould cavity with an inflow velocity of 0.9144 m/s after 0.1 second. The result (*Figure 4.48b*) obtained using the SOLA-VOF algorithm shows a similar trend and profile, even though the inflow velocity used was slightly higher (1.1 m/s).

Figures 4.49a to *4.49c* show the flow patterns after 0.25 second. All three methods show a similar trend and profile. The experimental result (*Figure 4.49d*) [Hwang and Stoehr (1987)] included here is meant to give some reference to how the fluid flow behaves in reality. The time at which the picture was taken is not known.

Figures 4.50a to *4.50c* show the liquid progressing to fill the cavity after 0.45 second. It is interesting to note that the SEA and SOLA-VOF have a similar profile while the liquid surface calculated by the MAC method spread much further towards the wall above the inlet.

Initial conditions:

Inflow velocity : 0.9144 m/s

Outlet (along the top edge of the mould): Pressure at outlet boundary is prescribed to zero.

Boundary conditions:

Walls: impermeable, isothermal, and non-slip or zero velocity at walls is assumed.

Gravity acting downward: 9.81 ms^{-2}

Other information:

Liquid volume flow rate : 3.484×10^{-2} m^3/s

Volume of mould cavity: 2.32×10^{-2} m^3

Expected fill-up time : 0.67 s

Table 4.16 Initial and boundary conditions for the academic case.

Relaxation factors:

Linear: pressure (0.8)

False time-step: u(0.05), v(0.05)

Time-step size: 0.001 s

Residual tolerance for velocities and pressure: 10^{-6}

Number of sweep per time-step: 40

Total number of time-steps: 900

Simulated real time: 0.9 seconds

Total run-time (wall clock): 5 hours (Sun Sparc IPX)

Table 4.17 Run-time parameters for the academic case.

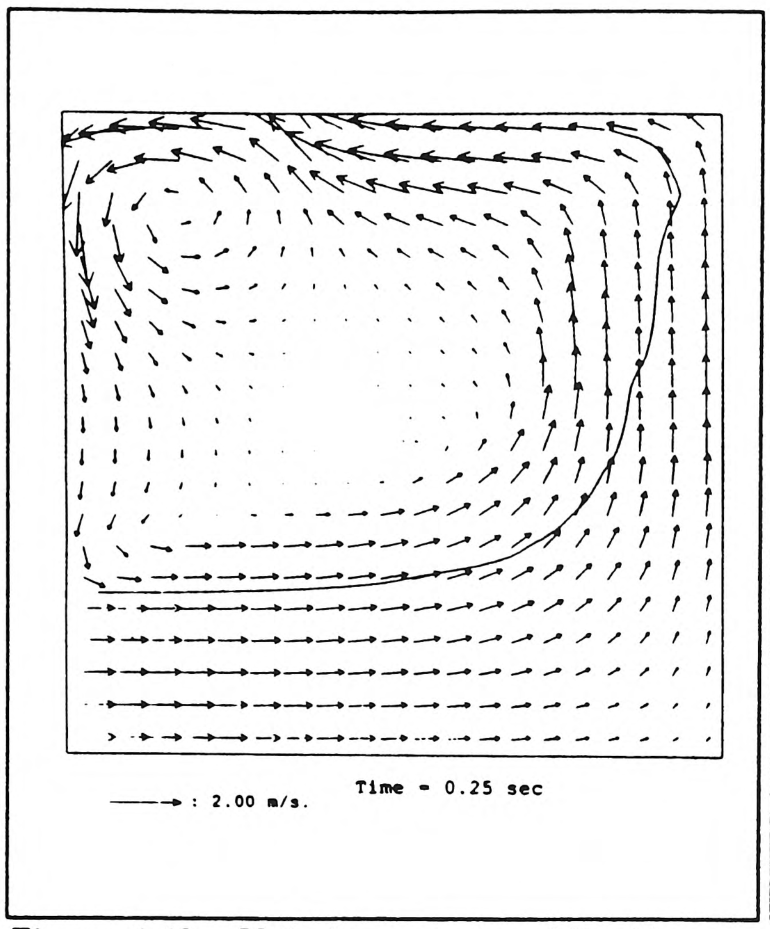


Figure 4.49a Velocity vectors and liquid surface after 0.25 second for the SEA case.

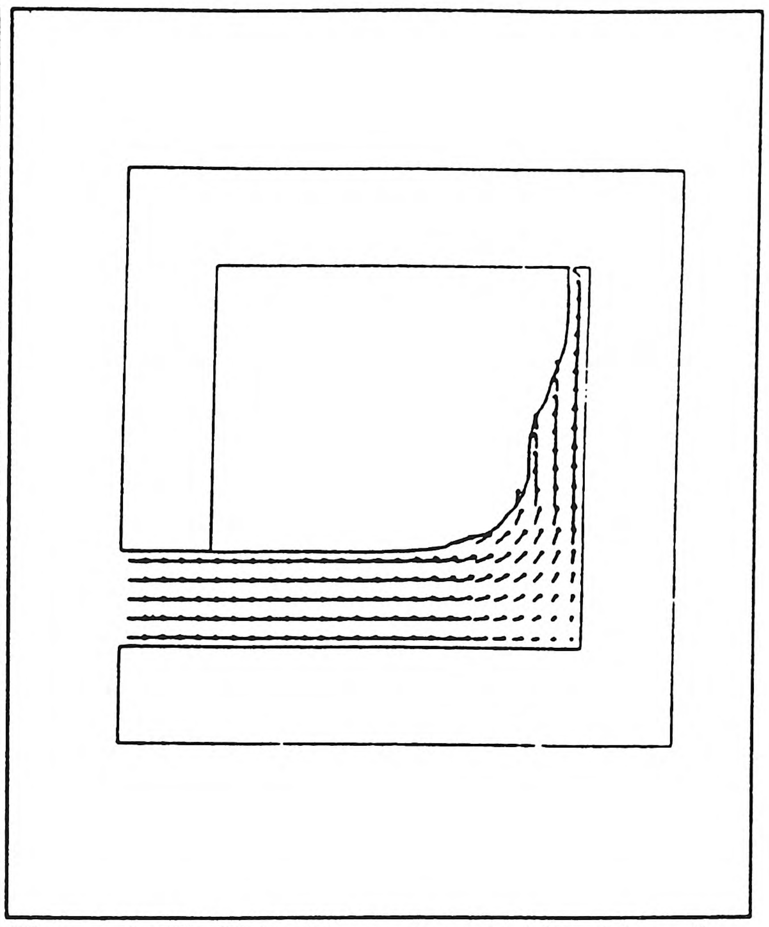


Figure 4.49b Velocity vectors and liquid surface after 0.25 second for the SOLA-VOF case.

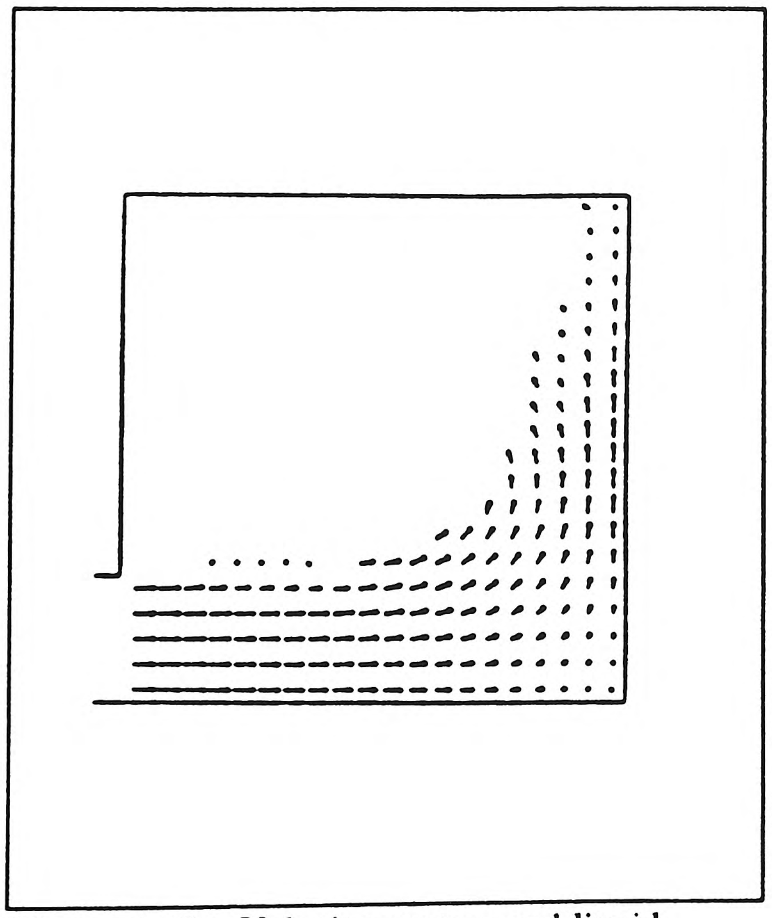


Figure 4.49c Velocity vectors and liquid surface after 0.28 second for the MAC case.

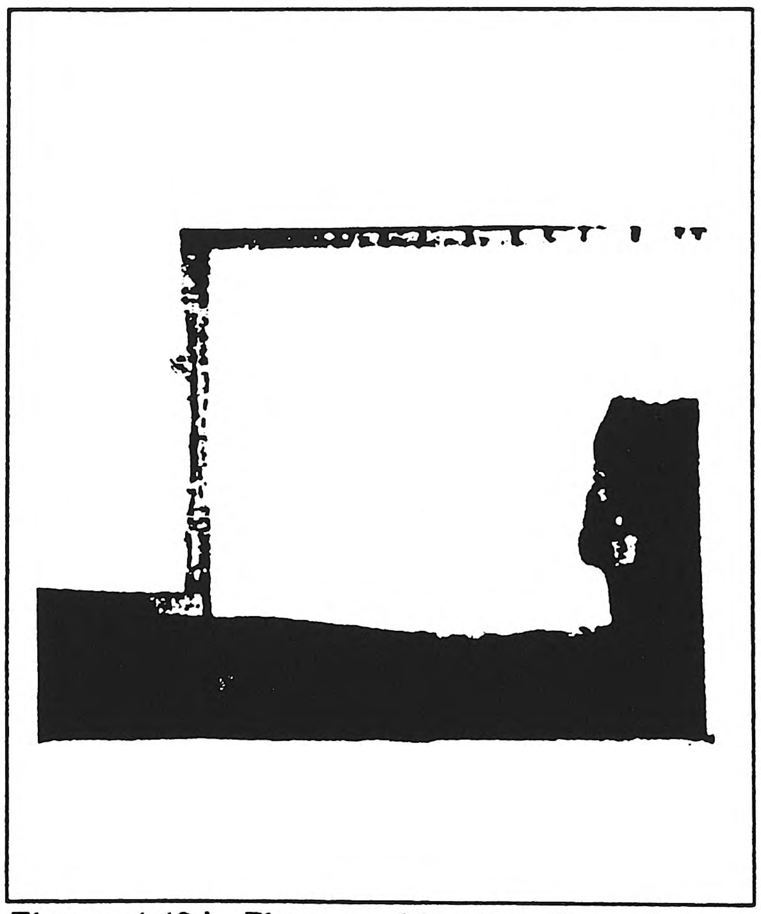


Figure 4.49d Photographic data for reference only, the exact time interval is not known.

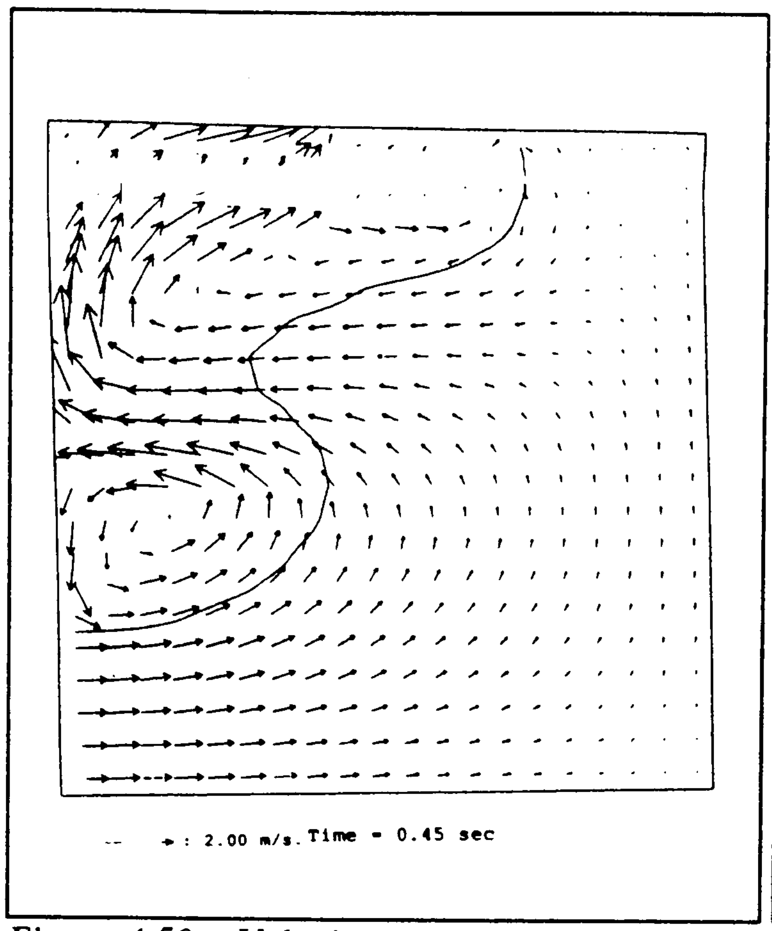


Figure 4.50a Velocity vectors and liquid surface after 0.45 second for the SEA case.

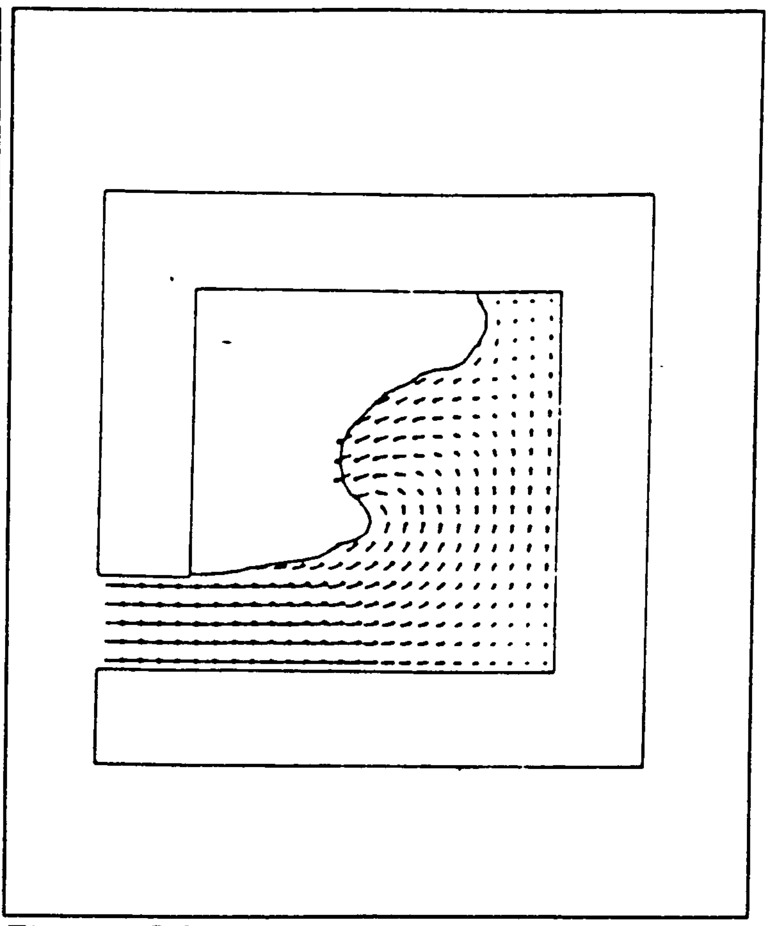


Figure 4.50b Velocity vectors and liquid surface after 0.45 second for the SOLA-VOF case.

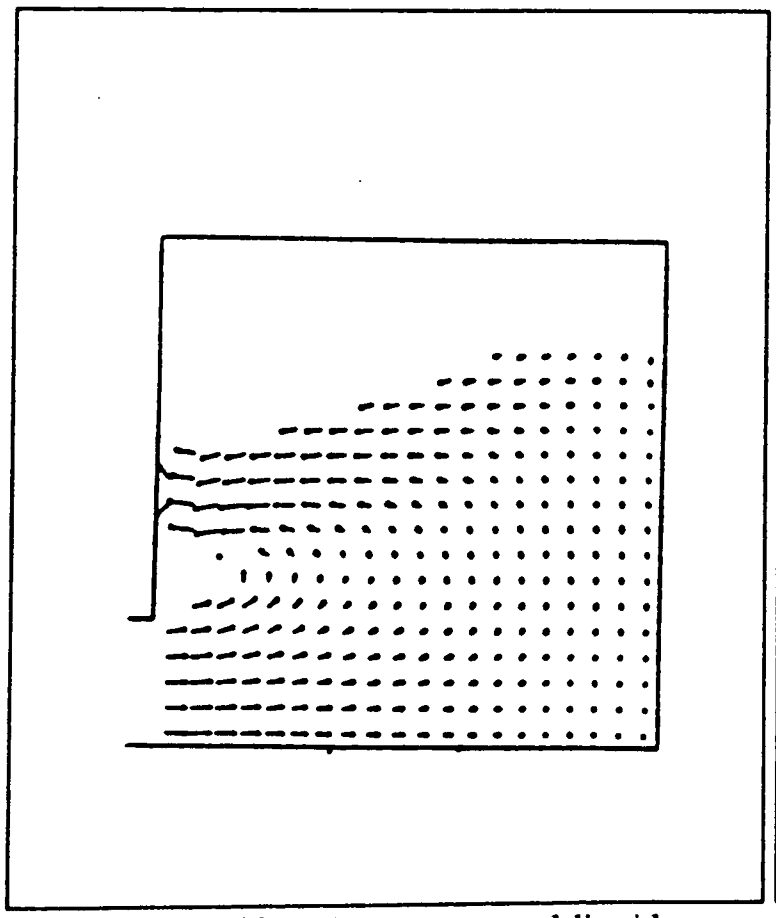


Figure 4.50c Velocity vectors and liquid surface after 0.46 second for the MAC case.

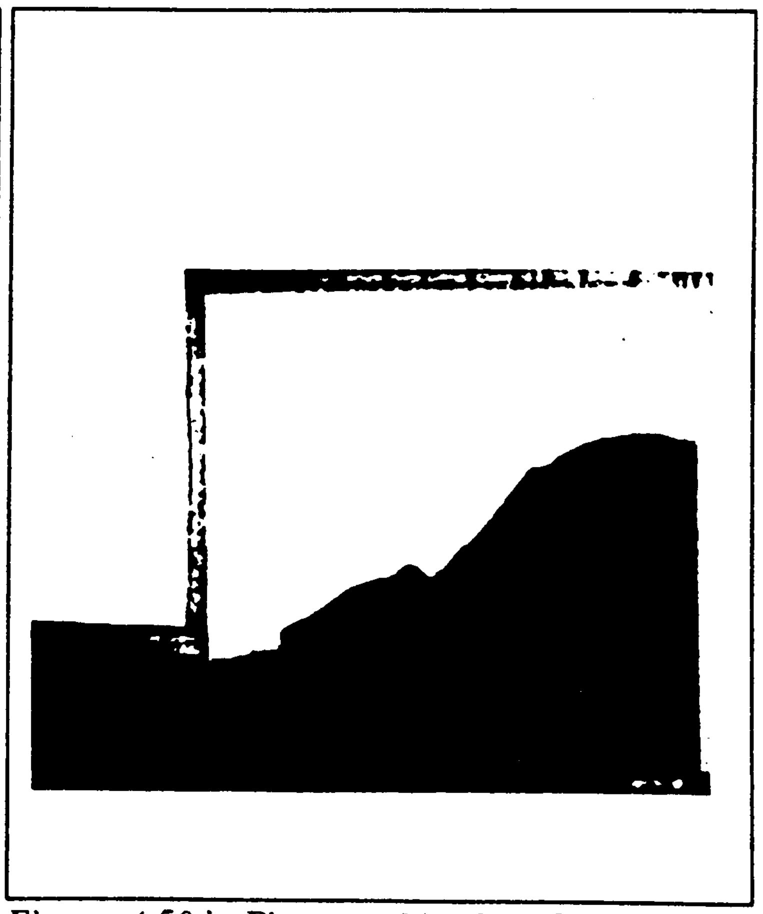


Figure 4.50d Photographic data for reference only, the exact time interval is not known.

Figures 4.51a and *4.51c* (after 0.55 second) show that liquid profiles from the three methods are quite different, especially the one calculated by SEA. In this case an air void is captured just above the inlet. Also, note that the timing for results obtained by the MAC method are slightly different to those indicated above. (The results obtained by the MAC method was taken after 0.54 second.)

Figure 4.52a and *4.52b* (after 0.6 second) shows that the mould cavity is almost full. The results from SOLA-VOF indicate that there is no entrapment of air. However, the air void captured by the SEA method during an earlier time step still remains. Although the experimental result (*Figure 4.52c*) included in this figure shows that no air void exists, air may be present in the form of bubbles.

Figure 4.53a (after 0.7 second) shows how the air is being squeezed out from the air void in the result obtained by the SEA method. On the other hand, the SOLA-VOF result (*Figure 4.53b*) shows recirculation at the same spot where the air void existed in the result from the SEA.

Figures 4.54 (after 0.8 second) and *4.55* (after 0.9 second) show the filling by SEA method. The mould is virtually full and excessive liquid is leaving the mould via the outlet at the top. The air void in within the cavity is diminishing in size.

4.6.4 Remarks for the Academic Case

The results produced by the SEA method for this case are in reasonable agreement with other simulated results. During the initial stage of the filling process, the SEA method matches very well with the results produced by the SOLA-VOF method. Later on, slight difference in the form of air bubbles, were appeared in the SEA results. Further work may need to investigate this discrepancy.

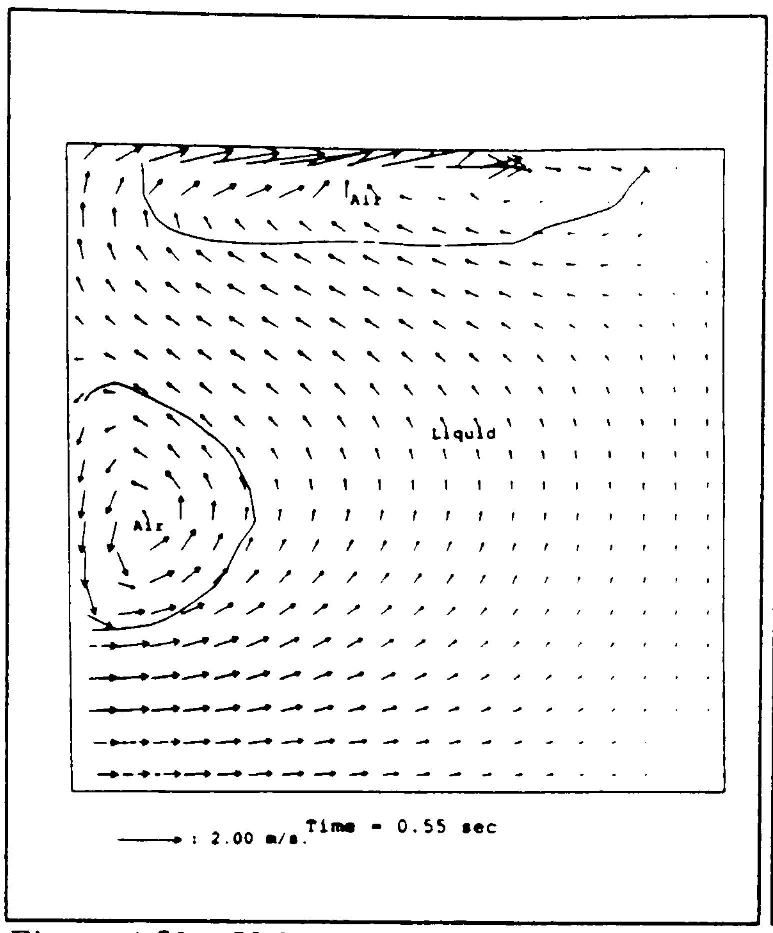


Figure 4.51a Velocity vectors and liquid surface after 0.55 second for the SEA case.

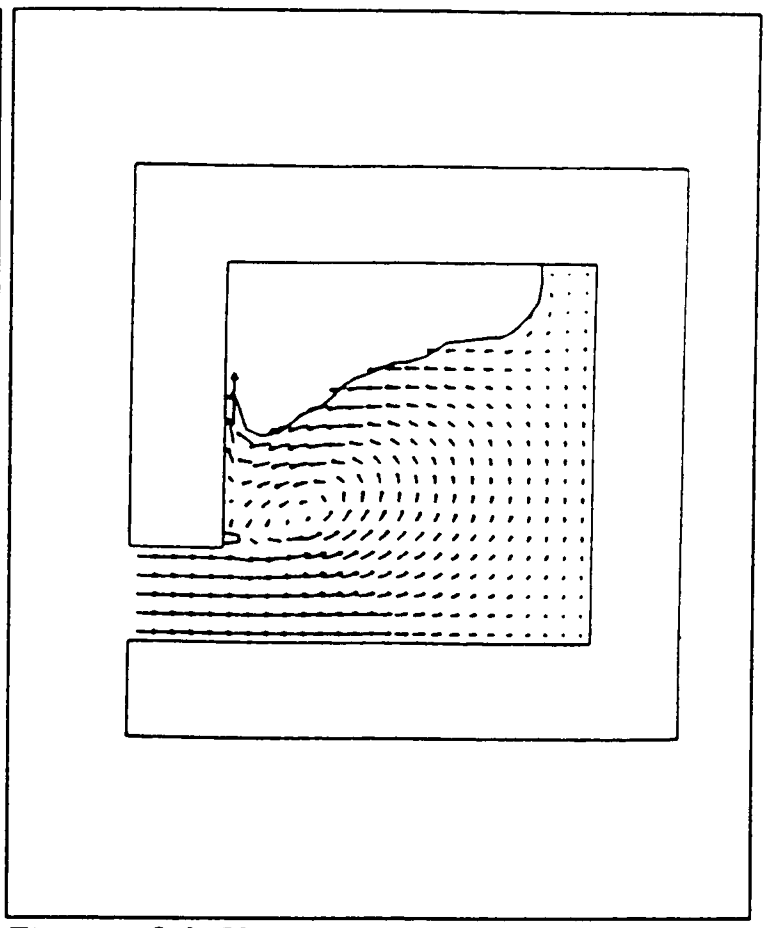


Figure 4.51b Velocity vectors and liquid surface after 0.55 second for the SOLA-VOF case.

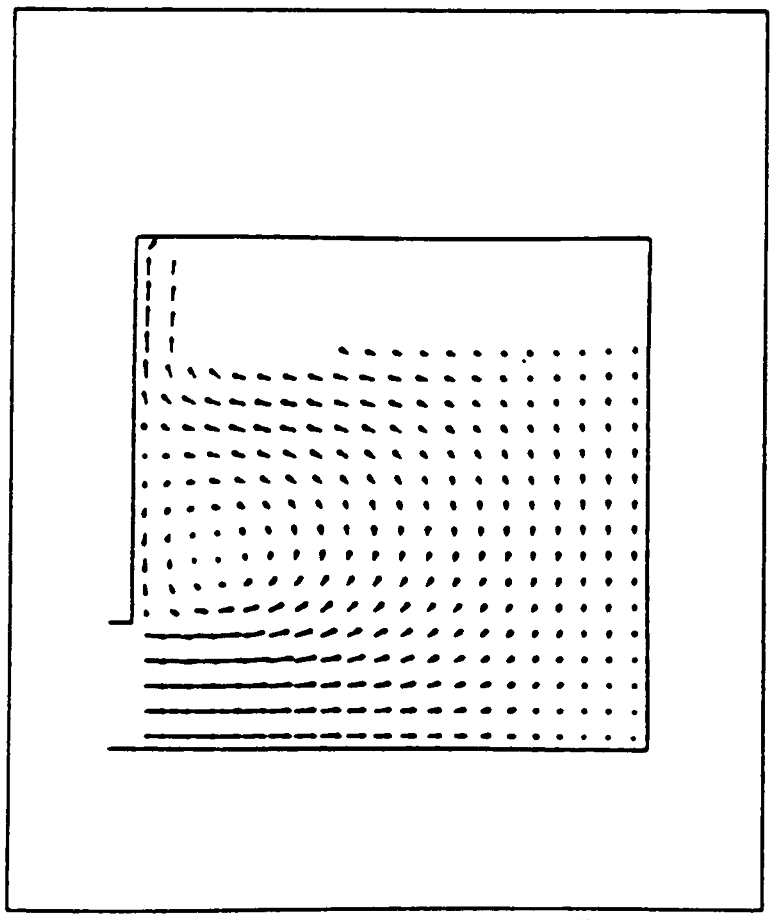


Figure 4.51c Velocity vectors and liquid surface after 0.54 second for the MAC case.

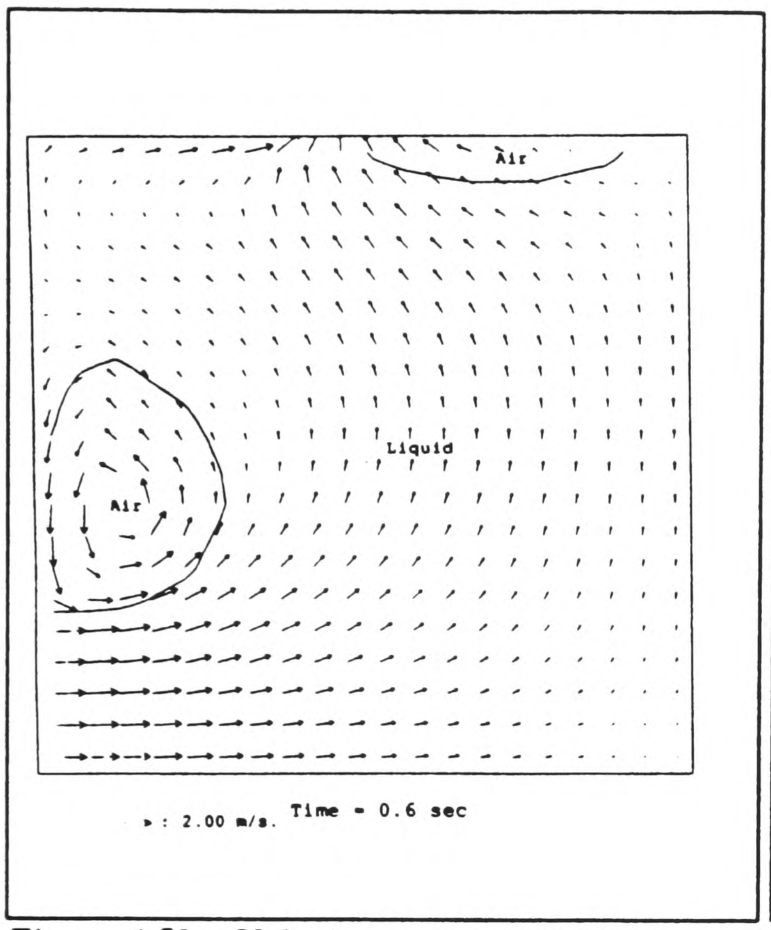


Figure 4.52a Velocity vectors and liquid surface after 0.6 second for the SEA case.

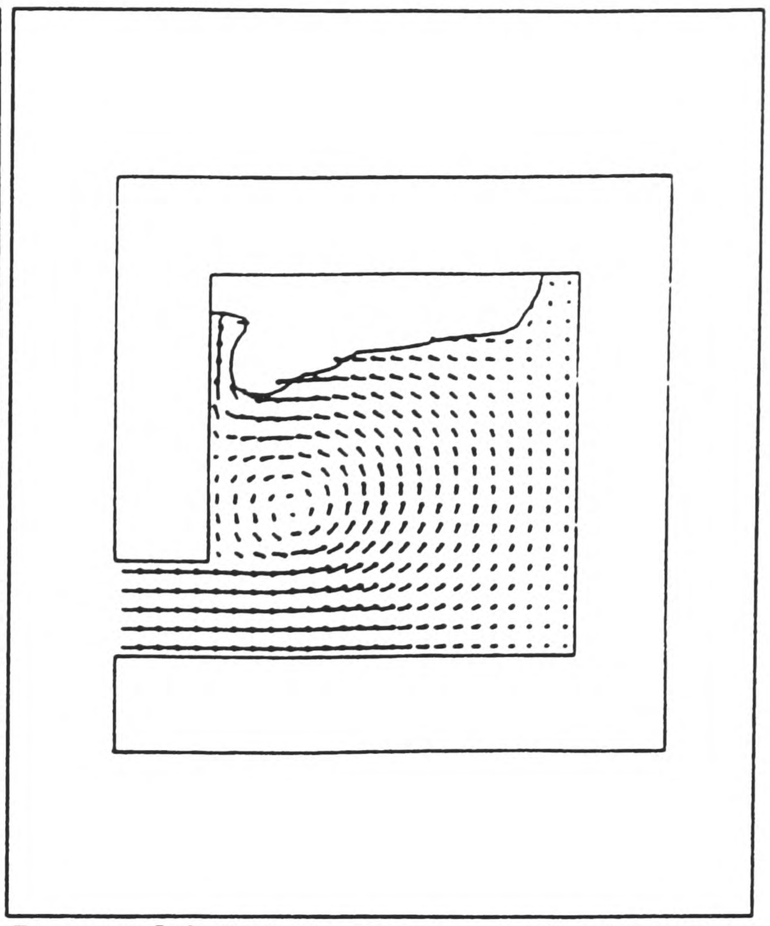


Figure 4.52b Velocity vectors and liquid surface after 0.6 second for the SOLA-VOF case.

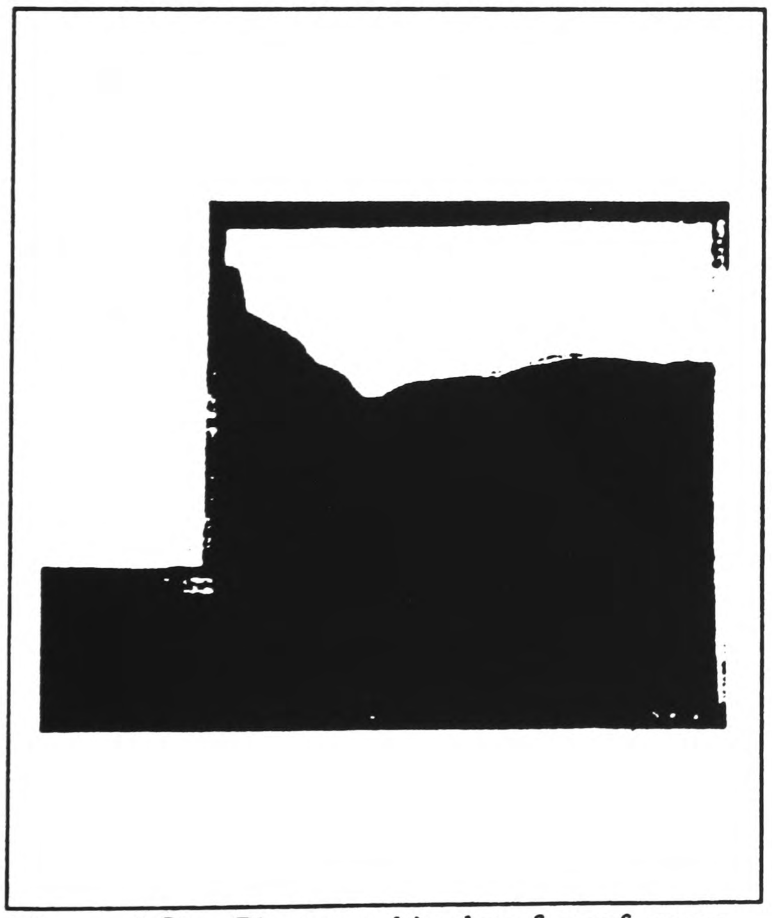


Figure 4.52c Photographic data for reference only, the exact time interval is not known.

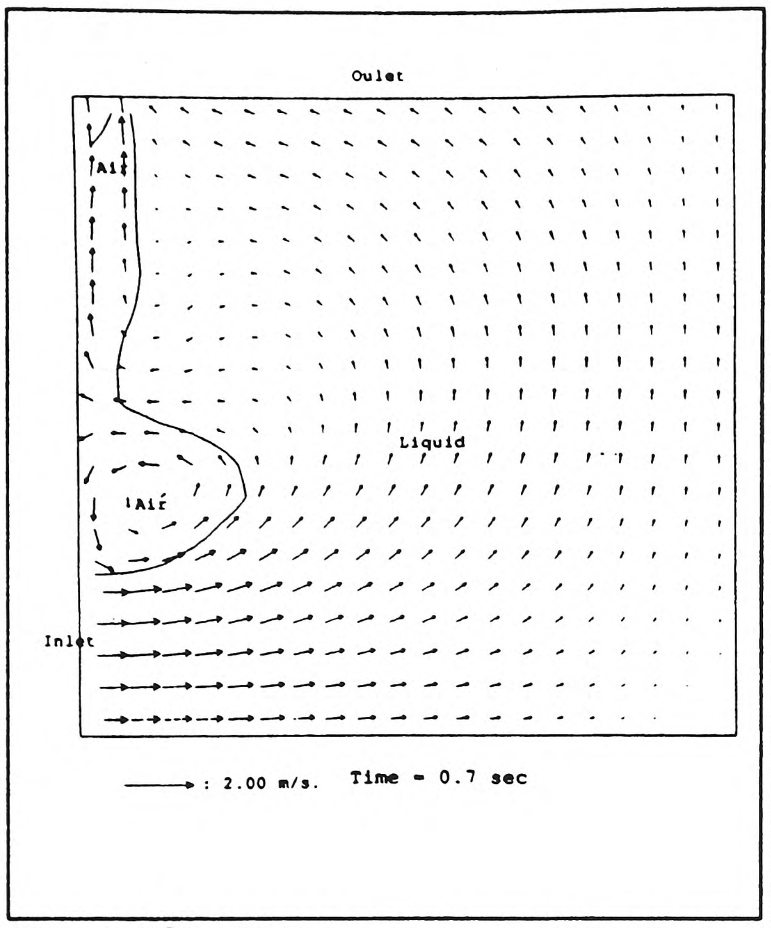


Figure 4.53a Velocity vectors and liquid surface after 0.7 second for the SEA case.

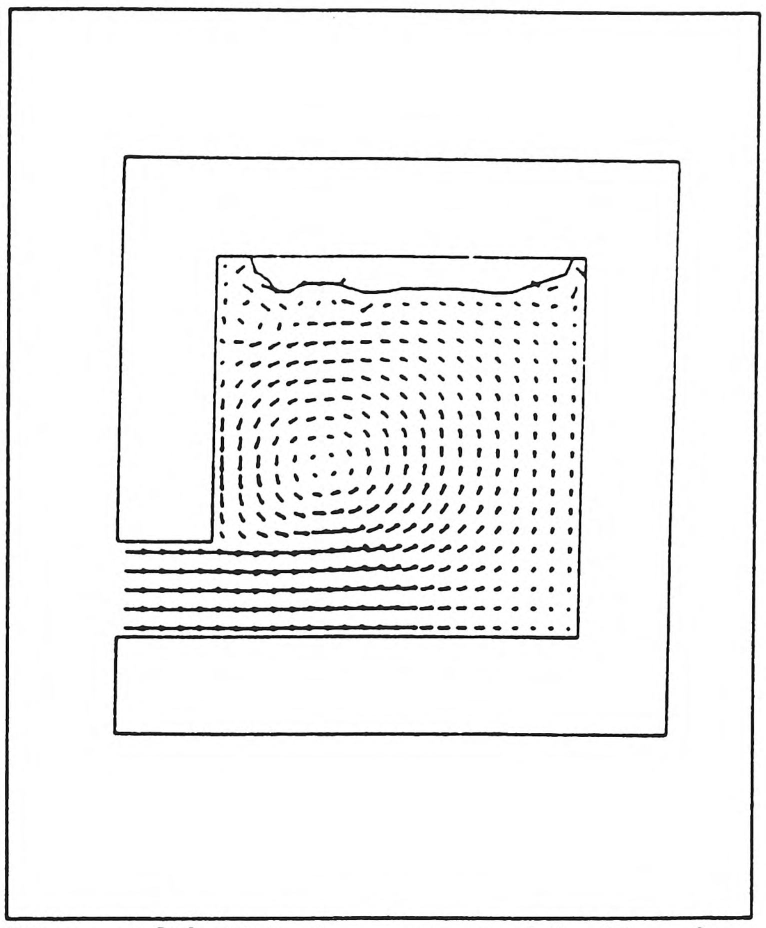


Figure 4.53b Velocity vectors and liquid surface after 0.7 second for the SOLA-VOF case.

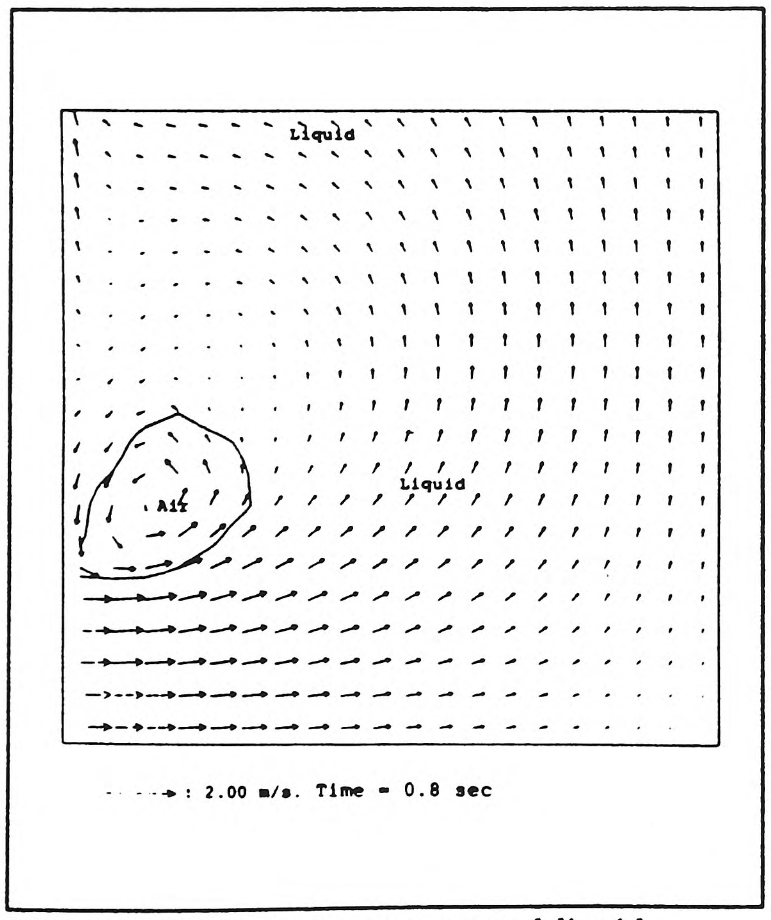


Figure 4.54 Velocity vectors and liquid surface after 0.8 second for the SEA case.

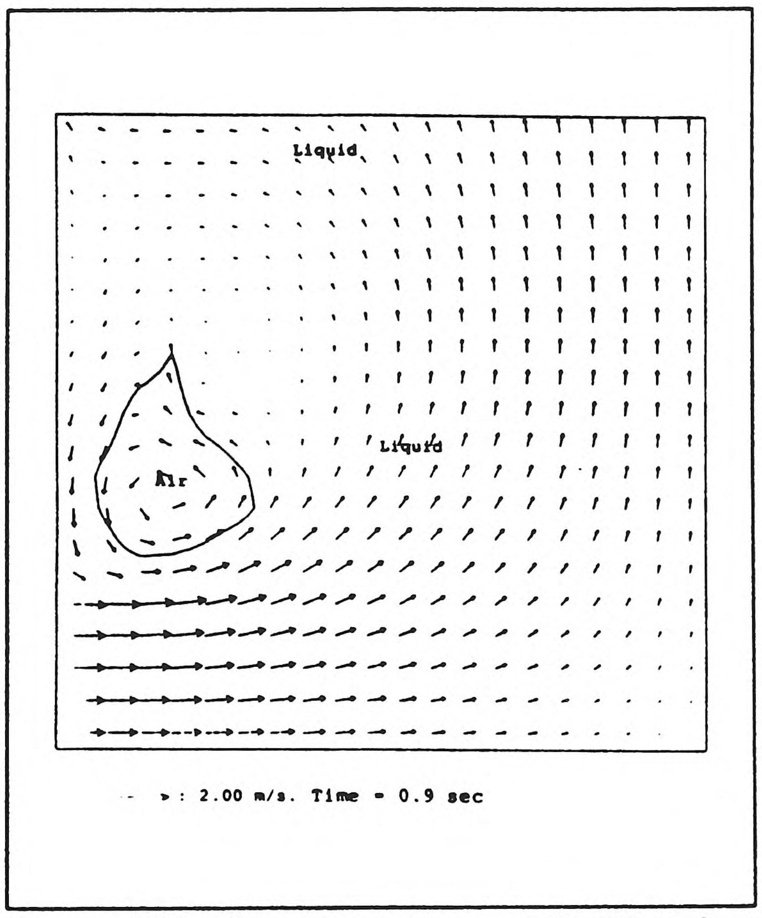


Figure 4.55 Velocity vectors and liquid surface after 0.9 second for the SEA case.

4.7 Closure

The capability of the three dimensional SEA filling algorithm in capturing complex wave movements and air entrapment in particular, has been illustrated through the five cases presented within this Chapter.

Chapter 5 *INCORPORATION OF THE FILLING MODEL INTO PARALLEL Harwell-FLOW3D*

5.1 Introduction

With the advances in VLSI (Very Large Scale Integration) technology used in the design and manufacturing of microprocessors, fast, complex and multi-channel microprocessors, such as the INMOS transputers and the Intel i860 vector processors, have emerged in recent years. These processors are all capable of performing hundreds or thousands of Mflops (Million floating-point operations per second) when running in *parallel.*

The performance of these microprocessors is impressive, but they all require software being written in a *specific way* such that the parallel features built within these microprocessors may be utilised. This specific way is known as *parallel programming*. Programs developed to run on the majority of current computer architectures, such as PCs and UNIX workstations, are most likely written to be executed sequentially by a single processor only. However, for parallel programs, more than one processor or *processing nodes* are expected to be executed simultaneously to give the desired performance. As a consequence, the degree of complexity within a parallel program is much higher than that of a serial program, and hence it is more difficult to implement. This is due to a number of factors, for instance, the availability of parallel computing hardware, software compiler and programming methodology. Especially the latter, which is subject to the type of hardware architectures (such as *shared memory* and *distributed memory*) to be used, and the nature of the application that the program is to provide.

Examples of large-scale CFD problems solved by using parallel computing architectures can be found in papers by Ierotheou and Galea (1992), Amato et al (1994), Daniels et al (1994), and Haberhauer (1994) respectively.

5.1.1 An Opportunity

Around the end of 1980's, S P Johnson at the University of Greenwich parallelised a version of the Harwell-FLOW3D code (AEA Technology) to run on a network of INMOS T800 transputers, a network of Intel i860's or any other distributed memory message passing systems [Johnson and Cross (1991), Johnson (1992)]. This code has been applied by Ierotheou and Galea to simulate fires in enclosures. The development of this parallel code has opened up an opportunity for the author to implement the filling model (described in *Chapter 3*) into the Harwell-FLOW3D code in order to investigate the following hypotheses:

1. *The filling model is built as a generic code module. Hence, it should be able to be implemented in other CFD codes without too much difficulty.*

2. *Running the filling model in parallel will improve its run-time efficiency.*

Due to the explicit formulation used in the SEA algorithm, the filling model is subject to small time steps (in the order of 10^{-3} second or smaller) imposed by the Courant criterion. Hence, this leads to high CPU time which may vary from hour(s) to day(s) on conventional machines, depending on the size of the problem tackled. This is both impractical and unacceptable to industrial users, such as the manufacturing engineers who are responsible to design the gating and runner system of moulds in foundries.

The rest of this Chapter is organised into the following subsections:

- *Section 5.2* describes the hardware and software available at the University of Greenwich for parallel computer processing.
- *Section 5.3* gives a general overview of the Harwell-FLOW3D CFD code.
- *Section 5.4* describes the strategy used in the parallelisation of Harwell-FLOW3D.
- *Section 5.5* describes the implementation of the filling model into the parallelised Harwell-FLOW3D.
- *Section 5.6* presents the results obtained by running the parallelised filling model on T800 transputer and i860-transputer systems respectively.
- *Section 5.7* gives a summary of the Chapter.

5.2 Parallel Computing Hardware and Software

The hardware and software used in the study of implementing the filling model onto the parallel Harwell-FLOW3D code are described as follows:

- INMOS T800 Transputer

The T800 transputer is a 32-bit RISC (Reduced Instruction Set Computer) microprocessor with 4-Kbytes of on-chip memory and can access up to 2 Gbytes of external memory. It also has a built-in floating point unit (FPU). Hence, it does not require a maths co-processor; and when it is running at 20 MHz a real performance of 0.4 Mflops can be realised. What makes a transputer differ from other serial microprocessors is that it has four 20 Mbytes/second communication links which enable it to be linked with four other transputers, see *Figure 5.1.* A variety of parallel distributed memory configurations can be set, such as a grid and hypercube.

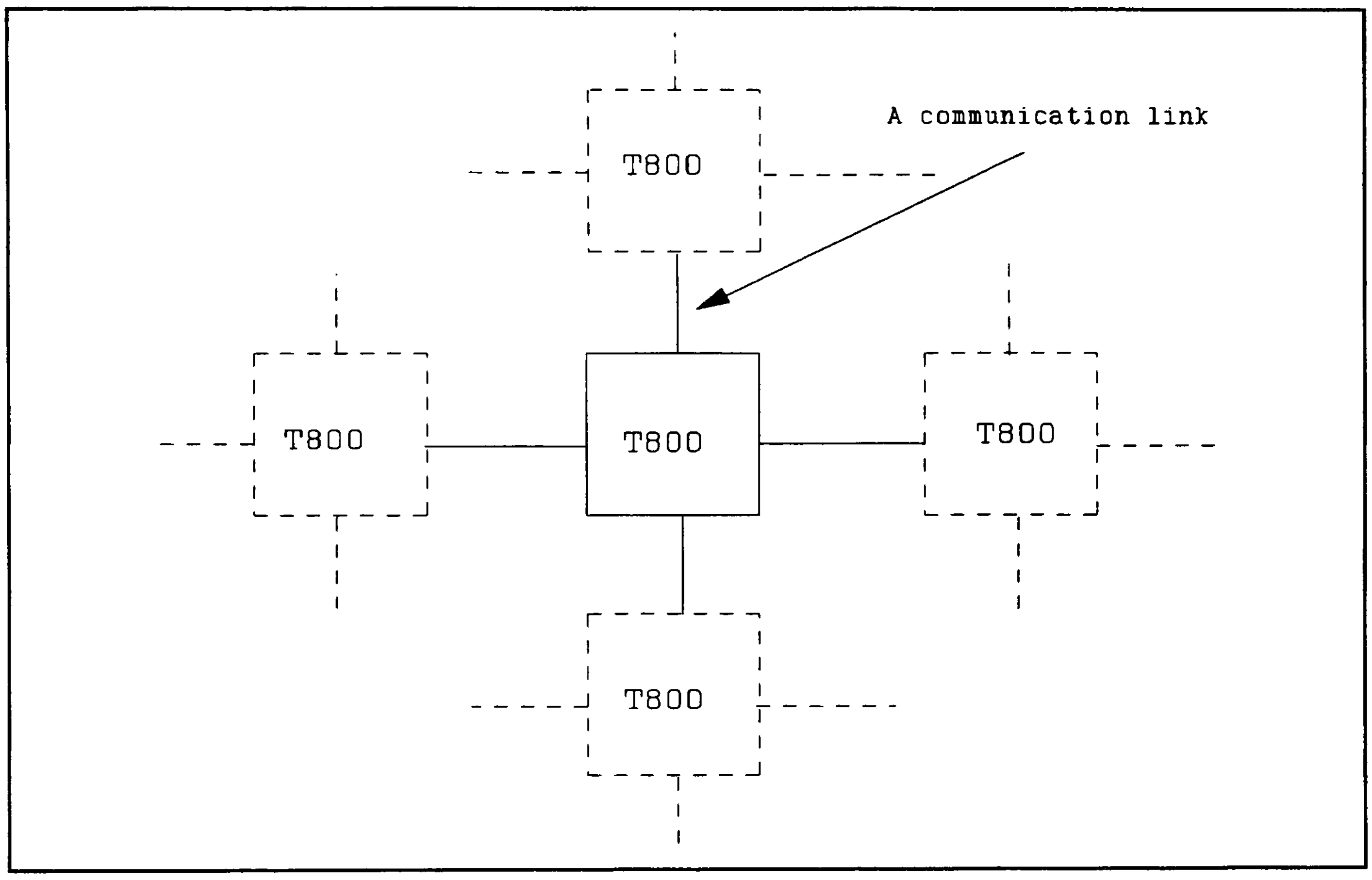


Figure 5.1 An INMOS T800 Transputer

• Hardware Platforms for T800 transputers

The transputer hardware at Greenwich are available to both PCs and Sun UNIX workstations. The PC-based hardware usually contains 16-bit ISA expansion cards which may accept up to four transputers each. Among these transputers, one is referred to as the *Master*, while the rest are *Slave* chips. Each transputer has 2 to 4 Mbytes of external memory, except the Master chip which has 8 Mbytes. These transputer cards can be attached to any IBM compatible 286, 386 or 486 computer. More than one card can be attached to a PC if more transputers are required to run in parallel. The Sun workstation based transputers are housed in a Transtech expansion unit which provides power and cooling for the transputers and their associated memory. Communications between Sun workstations (ie. the *host*) and the Transtech unit are via the Ethernet network.

• Intel i860

The Intel i860 is a 64-bit RISC microprocessor which contains about 1 million transistors. It comprises a CPU, three maths units, a three-dimensional graphics processor and 12 Kbytes of on-chip cache memory. A peak performance of 17 Mflops can be obtained when the i860 is running at 50 MHz. In real terms the performance of i860s is much lower, but it is still 10 to 13 times faster than a single T800 transputer (running at 20 MHz) depending on application. The designers at Intel claim that the i860 is a one-chip version of the Cray supercomputer [Hayes (1989)]. The reasons are that firstly the three maths-units on i860 are separate and can work on different problems in parallel. Together they can do integer mathematics, floating point additions and multiplications simultaneously. Furthermore, each maths unit is also designed for pipelining (see Johnson (1992) for a definition). The floating point adder and multiplier can be linked together for vector operations, in a way similar to Cray's vector-processing supercomputers. Hence, the i860 is classified as a vector-processing chip.

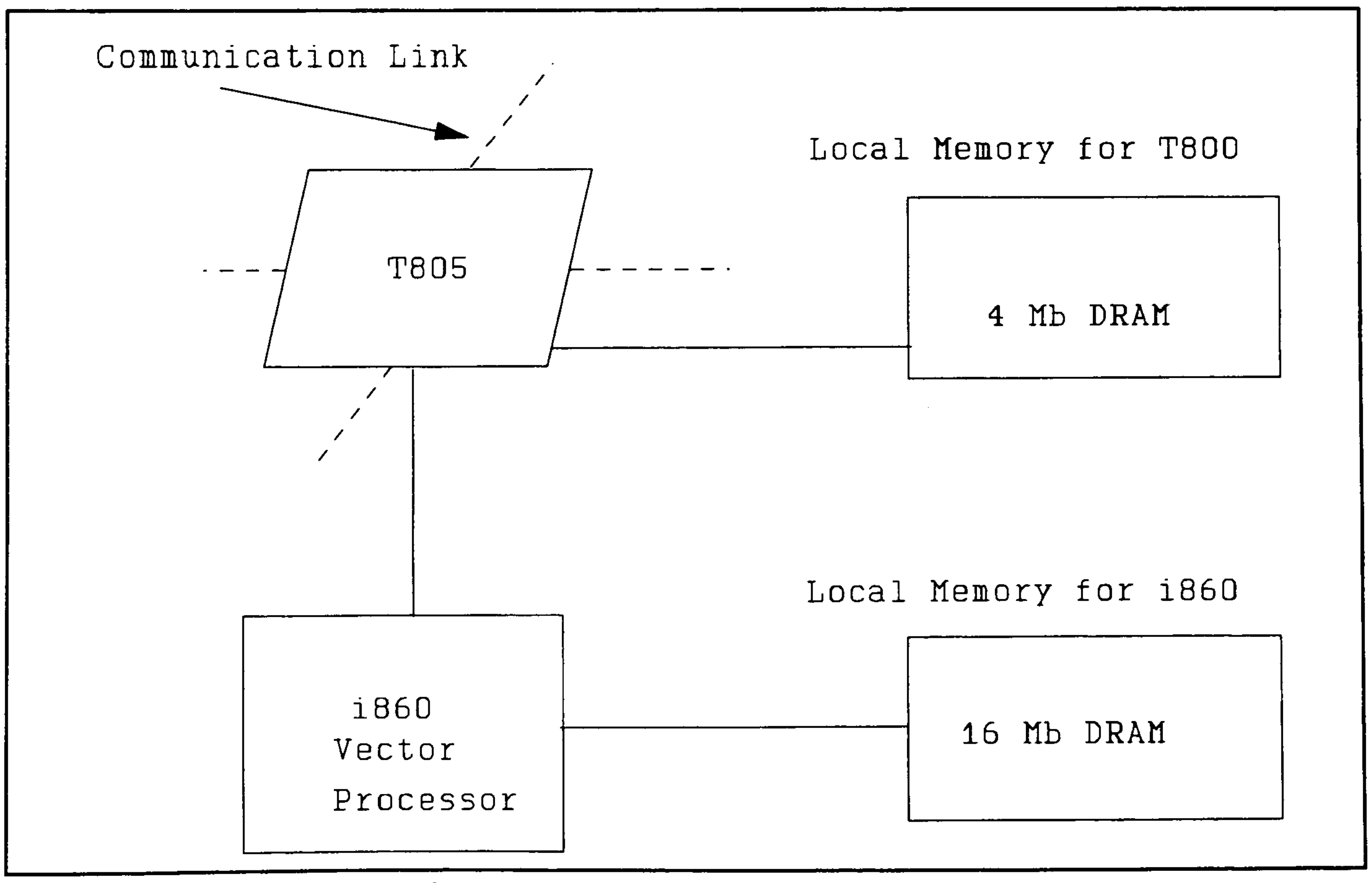


Figure 5.2 A i860 based transputer module

• Hardware Configurations for i860-transputer system

The i860 based system consists of 14 Transtech TTM110 transputer modules (TRAM) supplied by Transtech Parallel Systems. Each of these TRAMs consists of an Intel i860 chip running at 40 MHz, an INMOS T805 transputer (a variant of T800) running at 25 MHz, 16 Mbytes of 64-bit shared memory and 4 Mbytes of 32-bit local memory accessible only by the transputer, as shown in *Figure 5.2.*

The advantage of this configuration is to allow the i860 to use the transputer's communication facilities. Such that the configurations of the parallel network and the communications between the host and the i860's are the responsibilities of the T800s, whilst all number-crunching work is executed by the vectorised i860s. A single TTM110 TRAM was claimed by Transtech to have a floating point performance of 12.75 Mflops when running through FORTRAN Linpack benchmark tests [Transtech TM110 (1991)]. All these i860 TRAM modules are installed inside a Transtech expansion unit and connected to the Sun microsystem network.

• Software Compilers

A number of software compilers are available for transputer based hardware. These include, Occam, 3L-FORTRAN and C [3L Limited (1990)], and Portland Group (PG) FORTRAN [Portland Group (1992)] which are all designed to work with the transputer parallel architectures. Among them, Occam is especially designed by INMOS to work with transputers. Nevertheless, most scientific codes are developed in FORTRAN as it is the *de facto* standard language for scientific applications. For example, PHOENICS [CHAM Limited] and Harwell-FLOW3D [AEA Technology] are written in FORTRAN. Also from the perspective of code portability, Occam compilers may only work with transputer hardware, whereas FORTRAN compilers are widely available to most serial and parallel systems. Therefore, from the perspective of code portability and ease of code development, the Occam language was not selected as for developing the filling module, since Harwell-FLOW3D is written in FORTRAN. The 3L-FORTRAN is available to the PC and SUN platforms. It is used to develop code to run on transputers, whereas PG-FORTRAN is used by i860 processors. PG-FORTRAN is available on the Sun platform at Greenwich. In fact, both compilers offer identical standard FORTRAN features except the library routines for communications which are *processor dependent.* For example, the INMOS C-toolset, CStools or Parallel Virtual Machine (PVM) communication calls can be used on i860-transputer systems. And the 3-L communication calls work on homogeneous transputer systems.

5.3 An Overview of Harwell-FLOW3D

AEA Industrial Technology released the first version of FLOW3D in 1985. (Note that the FLOW3D package from AEA is referred to as Harwell-FLOW3D throughout this thesis, in order to distinguish it from a free-surface flow code also known as FLOW-3D of Flow Science Inc., USA.) Basically, Harwell-FLOW3D is a commercial, three dimensional, control-volume based, general purpose CFD package with the following capabilities:

- Steady/unsteady state simulations
- Incompressible or compressible flows
- Laminar or turbulent flows
- Heat transfer
- Multiphase capability.
- Chemical reactions and combustion etc.

And all these phenomena can be simulated within a Cartesian, cylindrical or body-fitted-coordinate (BFC) framework. For fluid flow simulations, Harwell-FLOW3D uses a family of SIMPLE [Patankar and Spalding (1972)] based velocity-pressure coupling algorithms, such as SIMPLE, SIMPLEC [Van Doormall and Raithby (1984)] and PISO [Issa (1985)], to ensure mass conservation within the calculation domain. As for numerical solvers, a wide choice such as Stone's strongly implicit [Stone (1968)], ICCG (Incomplete Conjugate Gradient) and line solvers are available. For differencing schemes, it offers hybrid, upwind, Quick and others. Furthermore, it has most of the usual boundary condition options that one would expect from a commercial code. These include adiabatic, isothermal, constant flux, no-slip/slip wall boundaries. See references [Jones I P et al (1985, 1986)] and [Harwell-FLOW3D user manual (1991)] for further details of this package.

Overall, Harwell-FLOW3D provides *similar* functionalities as PHOENICS (*Section*

3.6.1, Chapter 3) with a few exceptions. One of the exceptions is that a volumetric continuity equation, such as GALA [Spalding (1974, 1977)], was not implemented in Harwell-FLOW3D Release 2.2 (1988). Fortunately, the source code of Harwell-FLOW3D (release 2.2, 1988) was available for the parallelisation of the code itself [Johnson (1992)]. Hence, the author implemented GALA inside the source code to overcome this constraint.

The other fundamental differences between Harwell-FLOW3D and PHOENICS are listed below:

Harwell-FLOW3D	PHOENICS
• Non-staggered grid	• Staggered Grid
• Use dummy boundary cells	• No dummy boundary cells
• Use Rhie-Chow approximation (1982) for u, v, and w at cell-faces.	• No approximation required

Each of these differences between the two CFD products would have to be considered before the implementation of the filling module could begin. A straight *transplantation* of the filling module from PHOENICS to Harwell-FLOW3D would have proved to be a disaster. The implementation of the serial version of the filling module is described in *Section 5.5.*

5.4 Overview of the Parallelisation of Harwell-FLOW3D

The basic strategy applied to parallelise the Harwell-FLOW3D code is summarised in this section. Further details can be found in references [Johnson and Cross (1992), Johnson (1992)].

One way to parallelise a computer code is to have a *serial* version developed first and ensure that it is correct, before moving on to develop the parallel version. The serial version will then be used as a *benchmark* to compare the results generated by the parallel version. Hence, a serial version of Harwell-FLOW3D with the filling module was initially implemented.

The essence of the basic strategy in parallelisation, with respect to this investigation, is not to rewrite the code from scratch, but to adapt the original serial code with minimum code alteration. This strategy avoids the situation of re-inventing the wheel by wasting more time and resources. Also, more errors could be introduced into the parallel code and its subsequent maintenance would be greatly complicated as a result.

5.4.1 Configuration of Transputer Network

Various network configurations to link up a group of transputers are possible, these include chain or pipeline, hypercube, and grid. The simplest is the chain configuration as shown in *Figure 5.3*.

For this work the chain configuration was used. This was due to its simplicity in making physical and logical connections and thus good scalability, as well as due to the geometric parallelism which exists within Harwell-FLOW3D [Johnson and Cross (1991), Johnson (1992)].

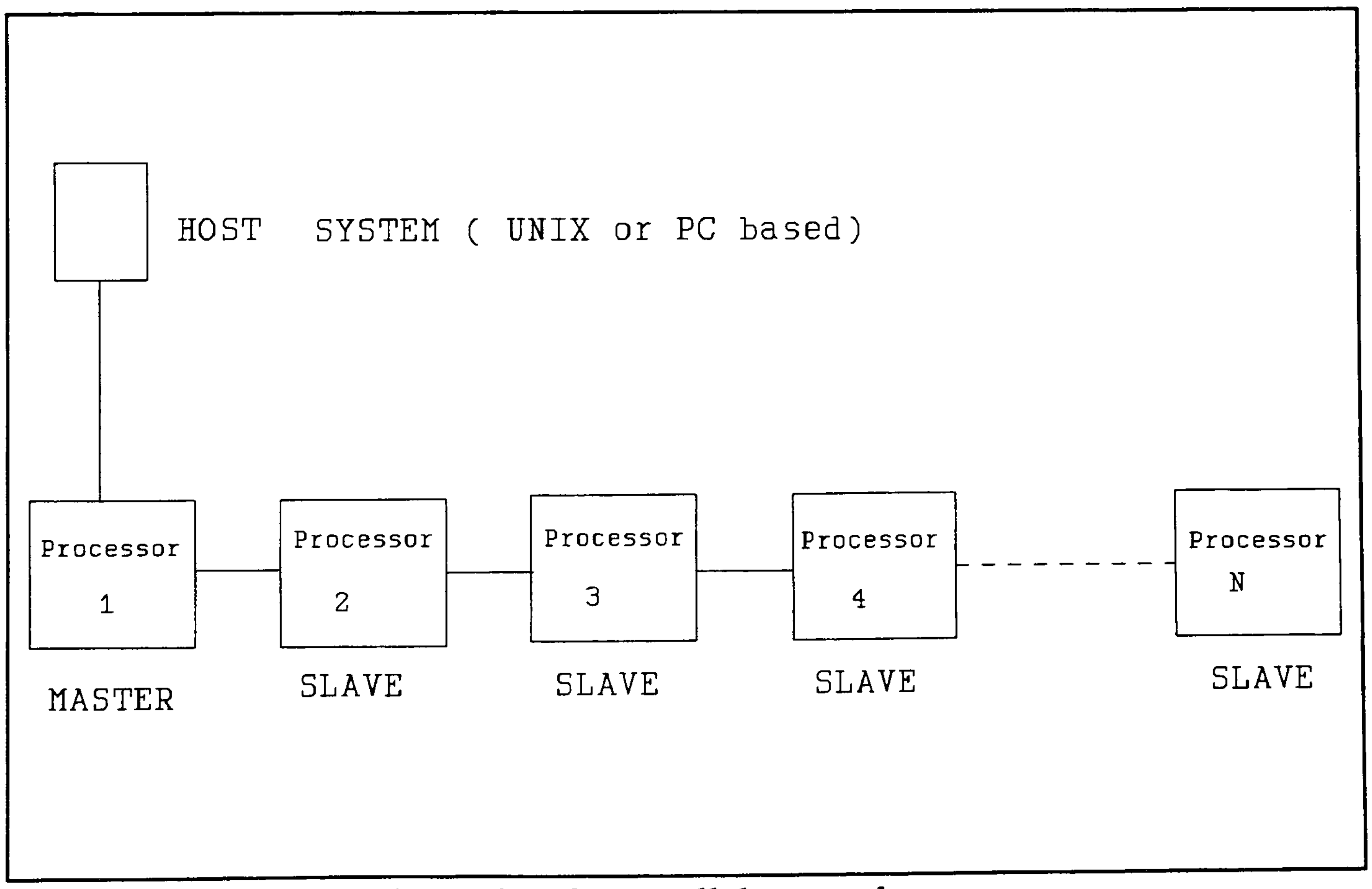


Figure 5.3 Chain configuration for parallel network

5.4.2 Decomposition of the Computational Domain

CFD codes have large data sets which are subject to almost identical mathematical operations, for example all CV cells are updated iteratively within a loop. This computational domain can be decomposed into a number of (3-D) cuboidal blocks, (2-D) rectangular blocks or (1-D) slabs of CV cells. The simplest way would be to partition data into slabs. In this case, only individual slabs of data (for example, the high and low faces in the z-direction) need to be communicated with other processors. Using four transputers for a computational domain with 20 slabs of data (or 20 cells in the z-direction) would require each transputer to have five slabs of data. With that the data is evenly distributed across the system. This is referred to as *load balancing*. However, when the computational domain is not evenly distributed among the transputers, a deterioration in run-time efficiency due to processor idleness may result.

The above domain decomposition strategy is applicable to codes which have structured, rectangular computational domains. Nevertheless, for codes with unstructured computational domains, such as ASTEC [AEA Technology] and ProCAST [UES Inc], the strategy for data partition is still effective, however the definition and calculation of the patterns are more complex.

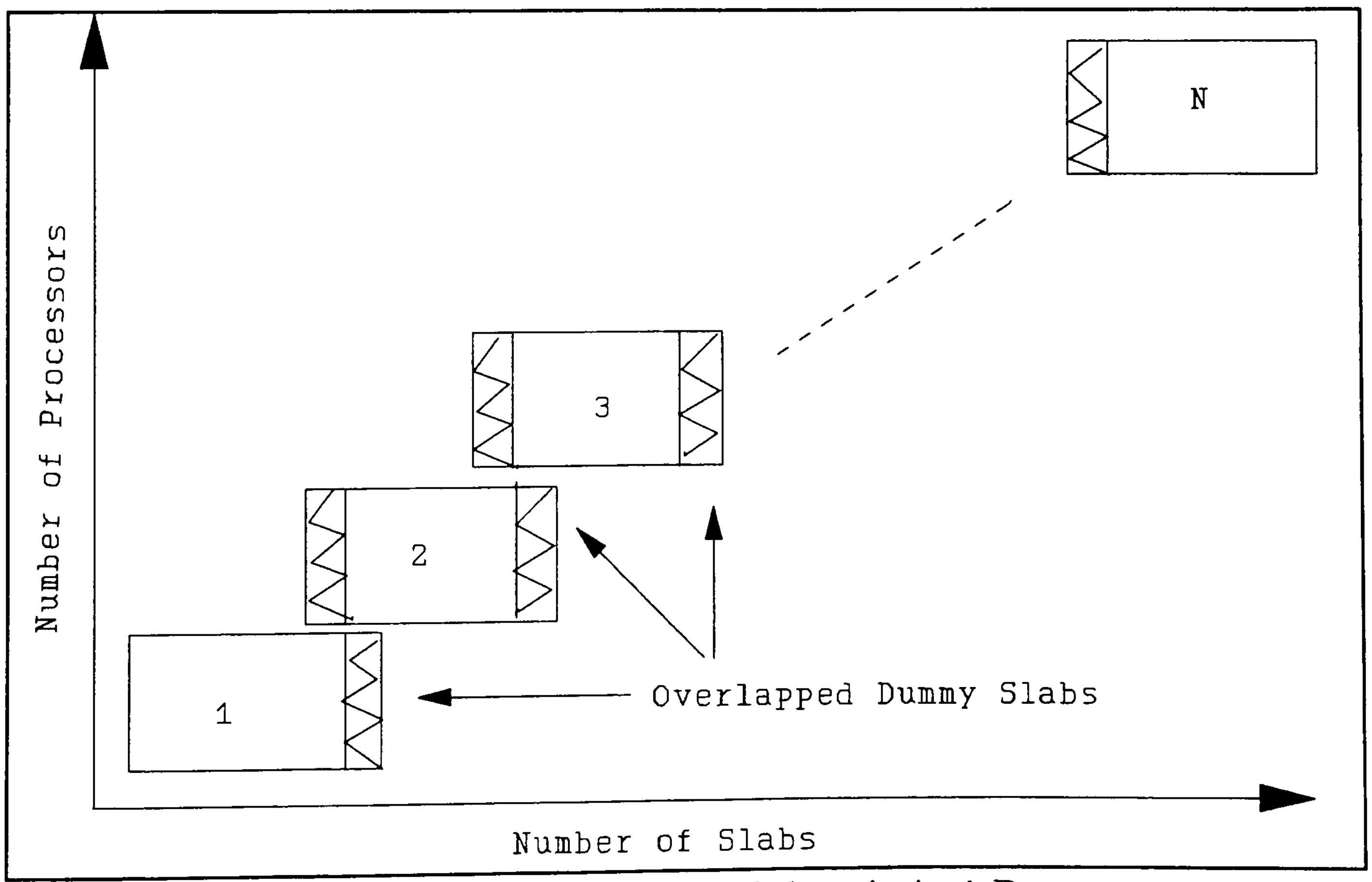


Figure 5.4 Decomposition of computational domain in 1-D

Hence, special algorithms such as the Extended Recursive Clustering Algorithm based on graph theory [Jones B W et al (1993)], may be needed to obtain the optimum data partitions.

Furthermore, the exchange of data between adjacent processor, is accomplished by having two dummy slabs or *overlap* areas (each of NX by NY cells, say); one at each end of the data block assigned to a processor. These dummy slabs of a processor are overlapped with those of adjacent processors as shown in *Figure 5.4*.

The information required by the neighbouring processors to complete, for example, a sweep are stored in these dummy slabs. The actual data-exchange is handled by subroutines which have been developed by Johnson (1992) based on various communication library routines.

5.4.3 Program Structure of the Parallel Version

The structure of parallel Harwell-FLOW3D can be sub-divided into two components; namely *master* and *slave(s)*. During run-time, the master version is down-loaded onto the master chip on the hardware while the slave version is distributed to other transputers on the board.

The master and slave versions share the same data and algorithmic structures and more or less the same subroutines, except where the master

1. handles all data input/output from the host machine (ie. the PC or Sun) to the transputer hardware;
2. allocates data partitions and sends all relevant problem definition information, such as material properties, locations of blockages and run-time parameters, to slave chips;
3. calculates global residuals for convergence.

5.4.4 Performance Measurement of Parallel Codes

To measure the performance of a parallel code the most quoted parameters are Speed-up (S_p) and Efficiency (E_p).

•Speed-up is defined as:

$$S_p = \frac{t_s}{t_p} \qquad (5.1)$$

where t_s is the execution times of a job running on a single processor using the serial version, and t_p is that running on p processors using the parallel code. S_p, in this case, shows how many times faster the parallel code is running on *p* processors than the serial running on a single processor. The results of parallel runs are shown in *Section 5.6.2.*

•Efficiency is defined as:

$$E_p = 100\left(\frac{S_p}{p}\right) \tag{5.2}$$

This parameter indicates how well the parallel code is utilising all available computing power, in terms of percentages (%). The maximum possible efficiency that a parallel code can achieve is not 100%, due to the overheads like communication time and processor idle time, which occur inherently in parallel codes. However, if large and extremely fast cache memory was available to buffer the data transfer between the transputer to local memory and/or to other transputers, then the efficiency could approach close to 100%.

5.5 Implementation of the Filling Module in Harwell-FLOW3D

The implementation of the serial filling module was done externally via the user routines provided by Harwell-FLOW3D in a way similar to PHOENICS. However, the implementation of GALA had to be done internally within the source code of Harwell-FLOW3D. Alterations were made in several subroutines which govern the pressure correction calculation of the momentum field. The equations used for GALA and the filling module are basically similar to those described in *Chapter 3.* So they are not repeated here.

Without exception, some modification to the host code are required when porting a model from one platform to the other. When evaluating the scalar flux ϕ at cell-faces, due to the non-staggering nature of Harwell-FLOW3D, interpolation was used to re-stagger the grid, so that velocity at cell faces could be obtained. Alternatively, the discretised equations for ϕ-flux evaluations could have been rewritten to account for the effect of a non-staggered grid.

Preliminary comparisons were made with PHOENICS, whereby the agreement was generally good. However, because the objective of this exercise is to examine the possibility of mapping the module onto parallel architecture, no detail studies were made between the results from the two codes.

Once the serial version of the filling module had been implemented, the parallel version was developed with technical assistance from the original developer of the parallel Harwell-FLOW3D code [Johnson (1992)]. The parallel version is divided into master and slave forms and the methodology used to implement them was described in the previous sections. *Figure 5.5* shows a data-flow structure diagram of the filling module attached to Harwell-FLOW3D.

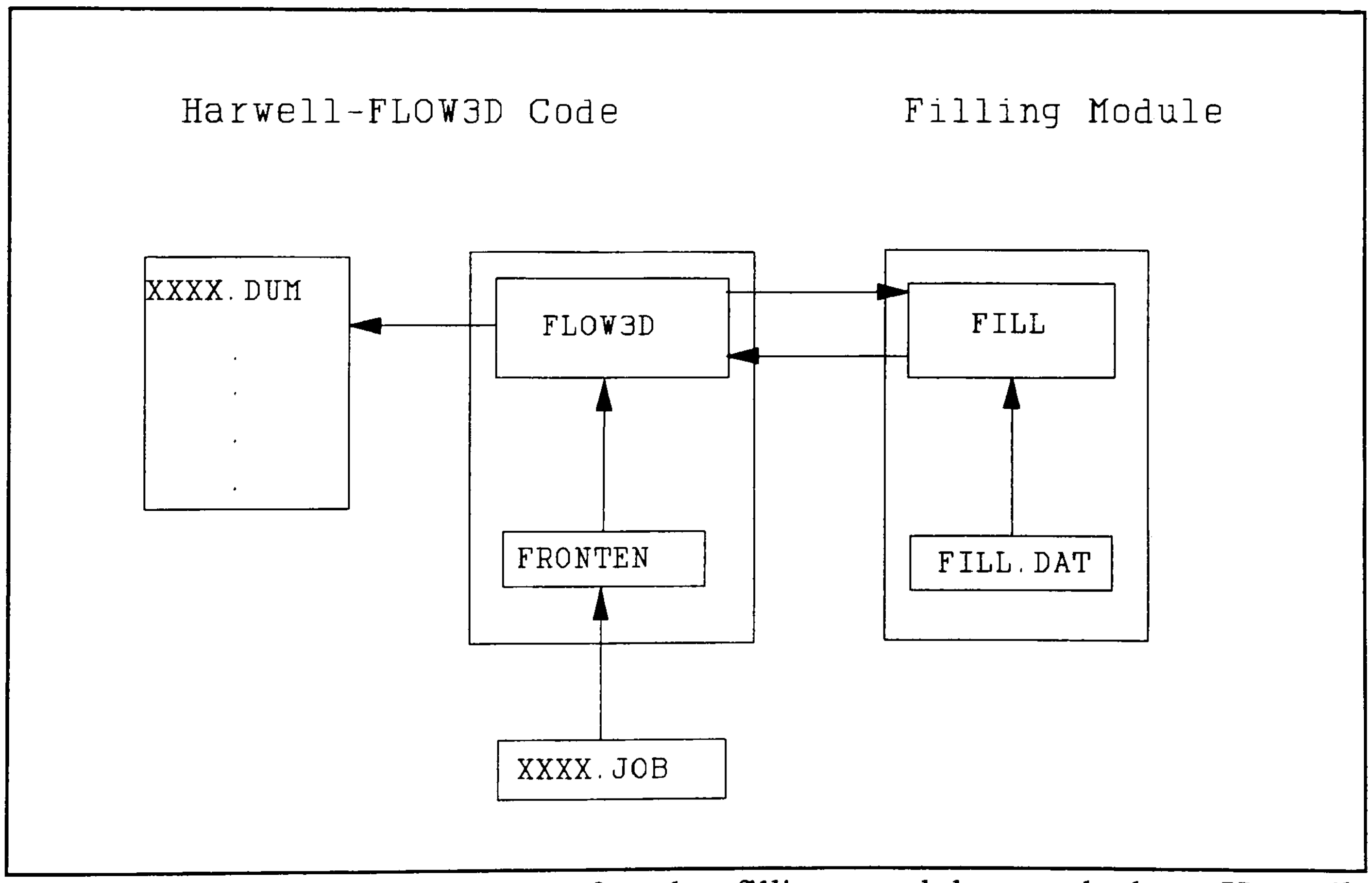


Figure 5.5 Structure Diagram for the filling module attached to Harwell-FLOW3D

This figure, shows a .JOB file which contains command language instructions for the problem specification pre-processor 'Fronten'. Dimensions of the domain, variables to be solved, boundary conditions, blockages, time step and other run time parameters can all be specified using this command language for Harwell-FLOW3D. However, data which is specific to the filling module and cannot be prescribed in the default .JOB file, are stored in a file called *FILL.DAT*. The results calculated by Harwell-FLOW3D are stored in file(s) with extension .DUM. A series of these .DUM files are produced when a transient simulation, such as mould filling, is carried out.

5.6 A Test Case: Filling of a Rectangular Container

Simulations of a thick-walled rectangular container filled with water performed on transputer and i860 based parallel hardware respectively, are presented here.

5.6.1 Problem Specification

The dimensions of the container are 35.5 x 14.2 x 3 cm (as only half of the container is modelled). The inlet and outlet are both situated at the top left and right walls respectively, as shown in *Figure 5.6.*

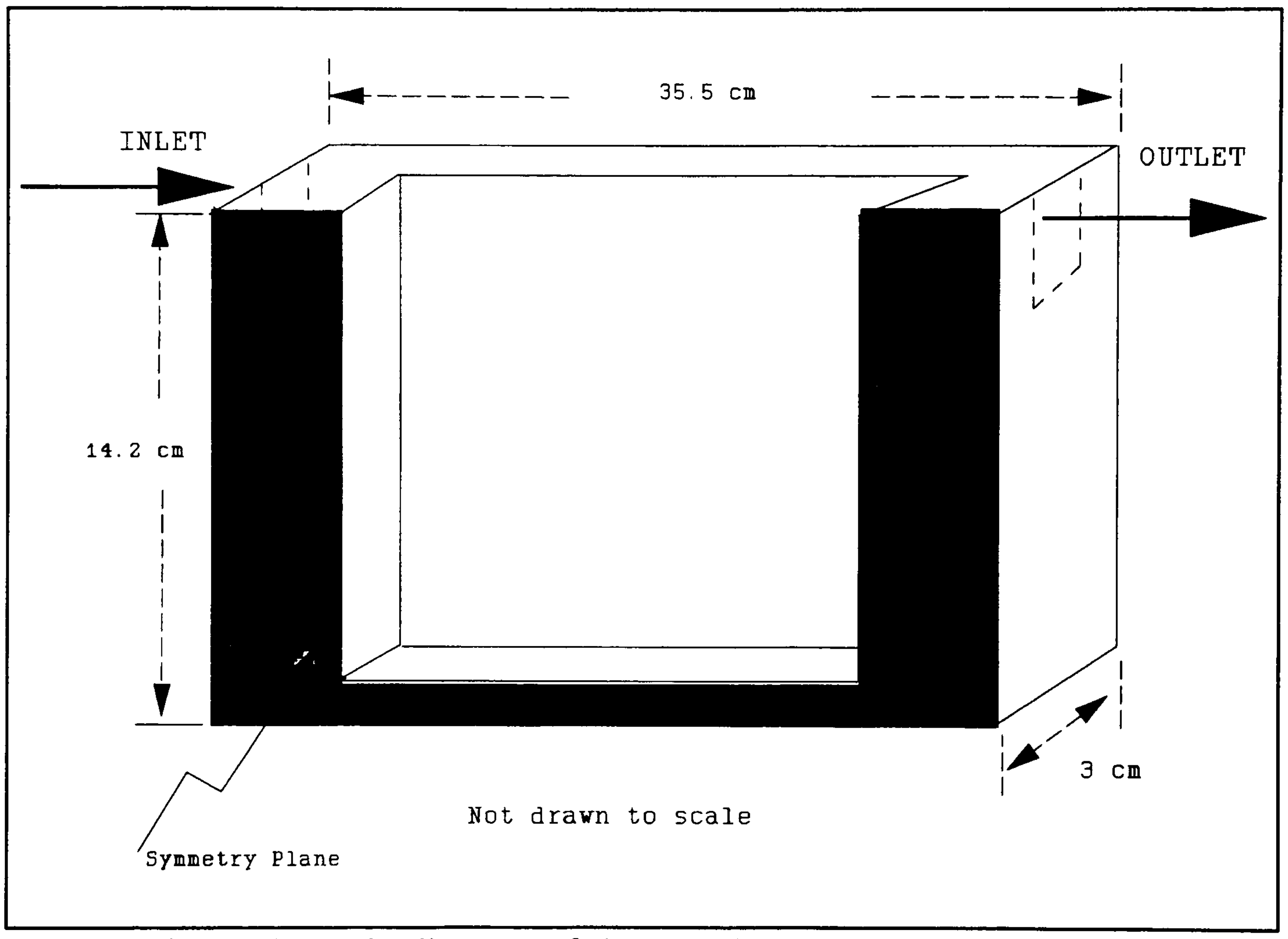


Figure 5.6 A schematic diagram of the container

The material properties (*Table 5.1*) and boundary conditions used in the simulations are shown as follows:

	Water	Air
Density, ρ (Kg/m^3)	1000.0	1.0
Kinematic Viscosity, ν (m^2/s)	10^{-6}	10^{-5}

Table 5.1 Material properties for the filling of a thick-walled container.

Boundary conditions:

• Inlet: a Dirichlet boundary condition.

Inlet velocity is set to 1 m/s, flowing in at the Down face (according to the coordinate system adopted by Harwell-FLOW3D) on the near end of the container with a surface area of 0.0006 m^2.

• Outlet: a Neumann boundary condition.

A constant pressure of 0 is set at the Up face on the far end of the container as shown in *Figure 5.6*, with the same face area as the inlet.

• Gravity of 9.81 m/s is acting downward.

• No-slip conditions were by default used on all internal walls, except the symmetry plane at the South face (ie. the front of *Figure 5.6*).

Expected filling time:

Note that about 60% of the total volume (0.00151 m^3) of the container is empty so this region is set aside as blockages. This gives a free volume of 0.4 x 0.00151 m^3 = 0.000605 m^3 to be filled with water. For a constant volume flow rate of 1 m/s x 0.0006 m^2 = 0.0006 m^3/s, half of the container will take about 1.0 second to fill up. This is a rough estimation of the filling time.

5.6.2 Results

Results for Cases I and II have been published in a conference proceeding [Galea et al (1993)].

Case I.

A medium mesh of 22 x 13 x 40 = 11440 cells was used to simulate the filling of this container for 0.8 seconds in real time. The simulations were running on one and four i860-transputer based parallel nodes on the Sun system, To ensure the Courant criterion was satisfied 800 time steps of 0.001 second were used. In addition, each time step involved 21 iterations to ensure convergence.

Figures 5.7 to *5.14* depict clearly the formation of air cavities and wave actions which occurred when the moving liquid fronts interacted with walls. The isosurface of the liquid (water) at ϕ=0.5 is plotted. After 0.1 seconds, water has reached one third of the side wall and in contact with the base of the container. An air cavity is formed by the lower left hand corner of the container. At 0.2 second, the liquid front has advanced towards the end of the container and it is collapsing under its weight. Also, more liquid has reached the base of the mould. A large air cavity is formed at the middle of the back wall of the mould after 0.3 seconds. At 0.4 second, the water has reached the far end of the mould and splashed on the end wall. The small cavity seen earlier (0.1 and 0.2 second) has been filled completely, and the large cavity at the middle of the back wall is getting smaller. The mould is gradually filled up from 0.5 to 0.8 second.

The overall CPU time for the simulation running on four i860s was about 11 hours, see *Table 5.2* below.

Number of I860s	time (hours)	speed-up	Efficiency
1	23.5	-	-
2	14.9	1.58	79.0%
3	12.8	1.83	61.0%
4	11.3	2.10	52.0%

Table 5.2 11440 cells, 0-0.8 second, running on i860-transputer system

Figures 5.15 to *5.17* depict CPU times, speed-up and efficiency. A CPU time from 23.5 hours on a single i860 node down to just over 11 hours on four i860 nodes represents a reduction of over 50%. The speed-up and efficiency graphs of the 11440 cells case (in *Figures 5.16* and *5.17* respectively), indicates a marked degradation in performance. In other words, running on more processors will not necessarily give a linear gain in performance as one would ideally expect.

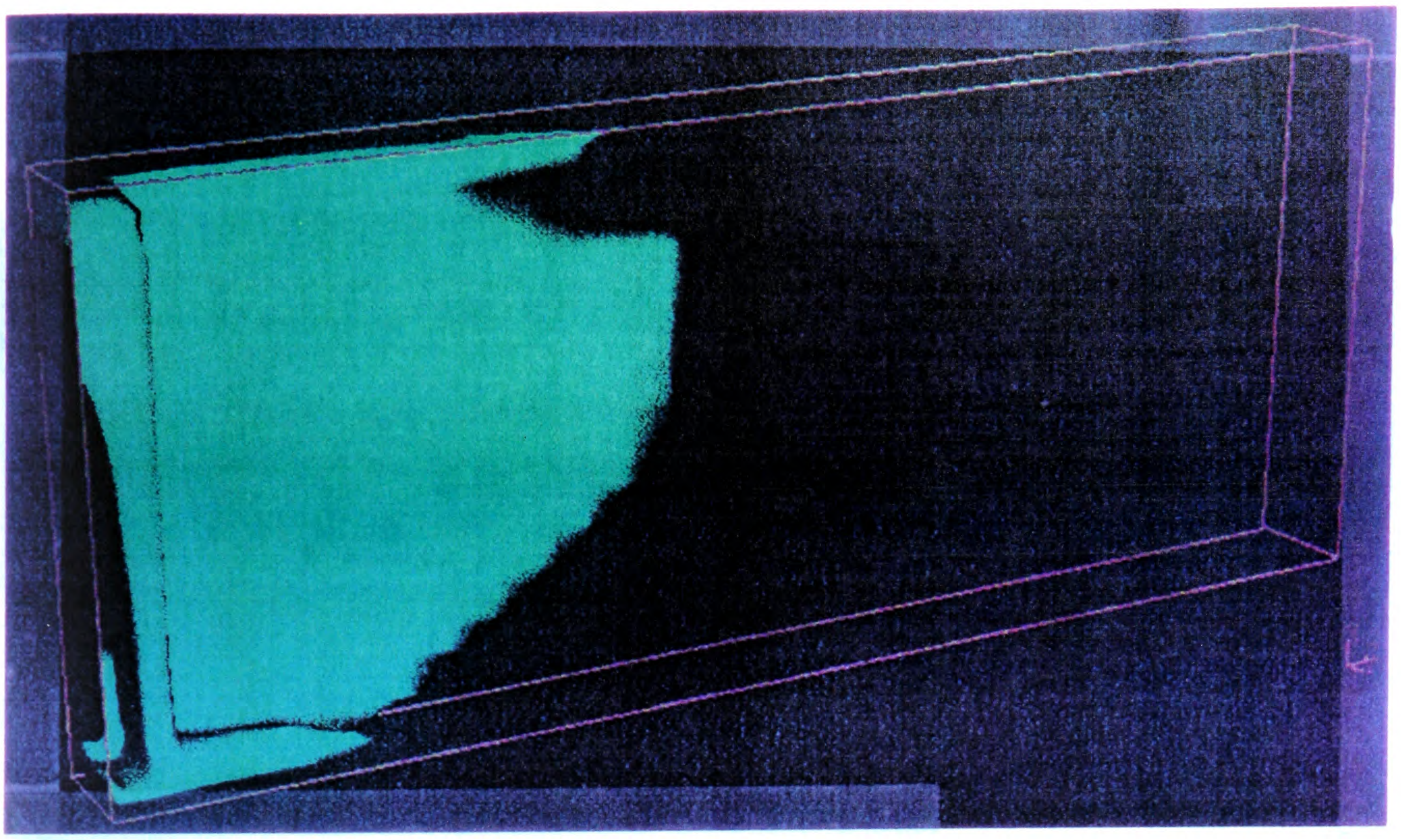


Figure 5.7 Visualisation of the liquid free surface after 0.1 second.

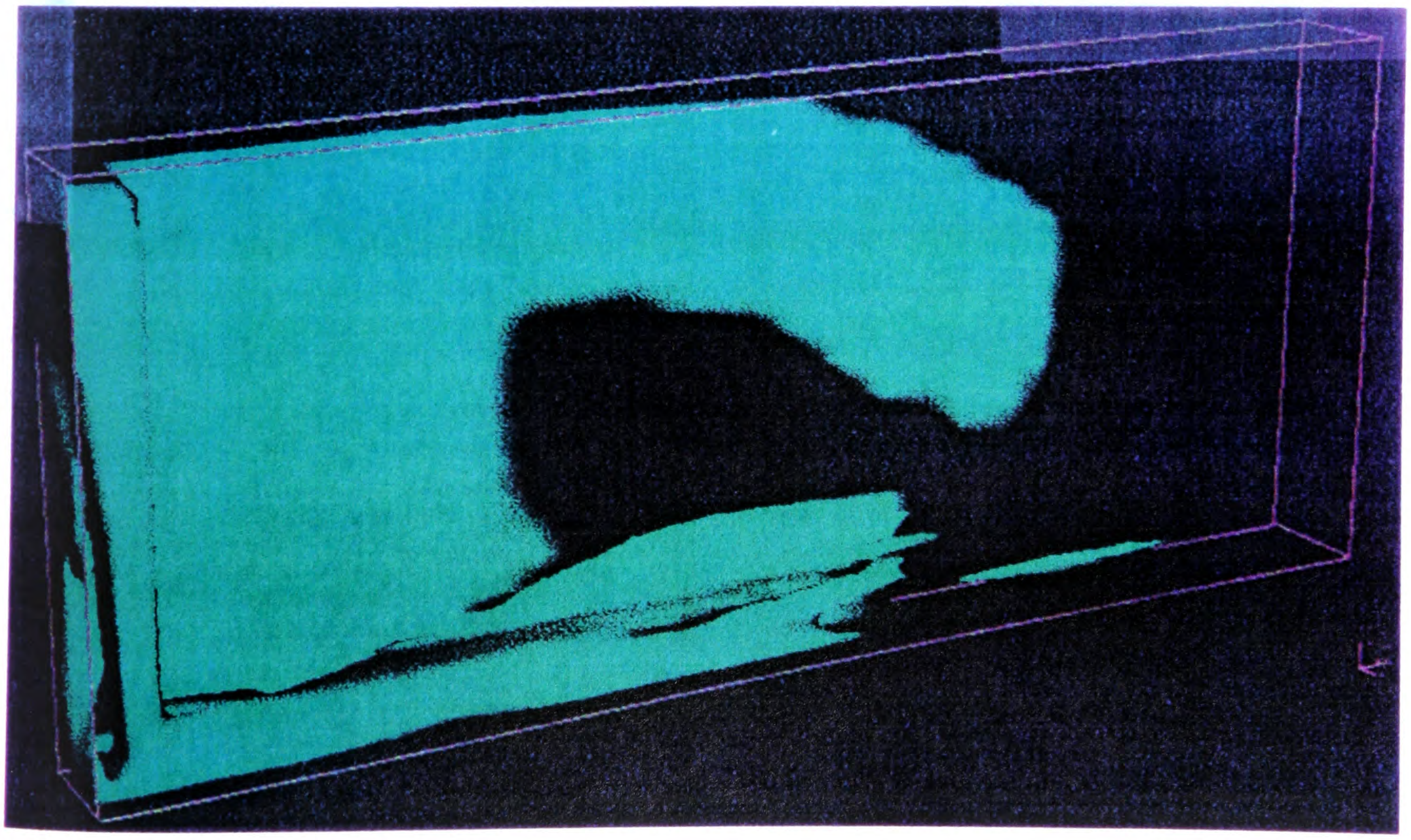


Figure 5.8 Visualisation of the liquid free surface after 0.2 second.

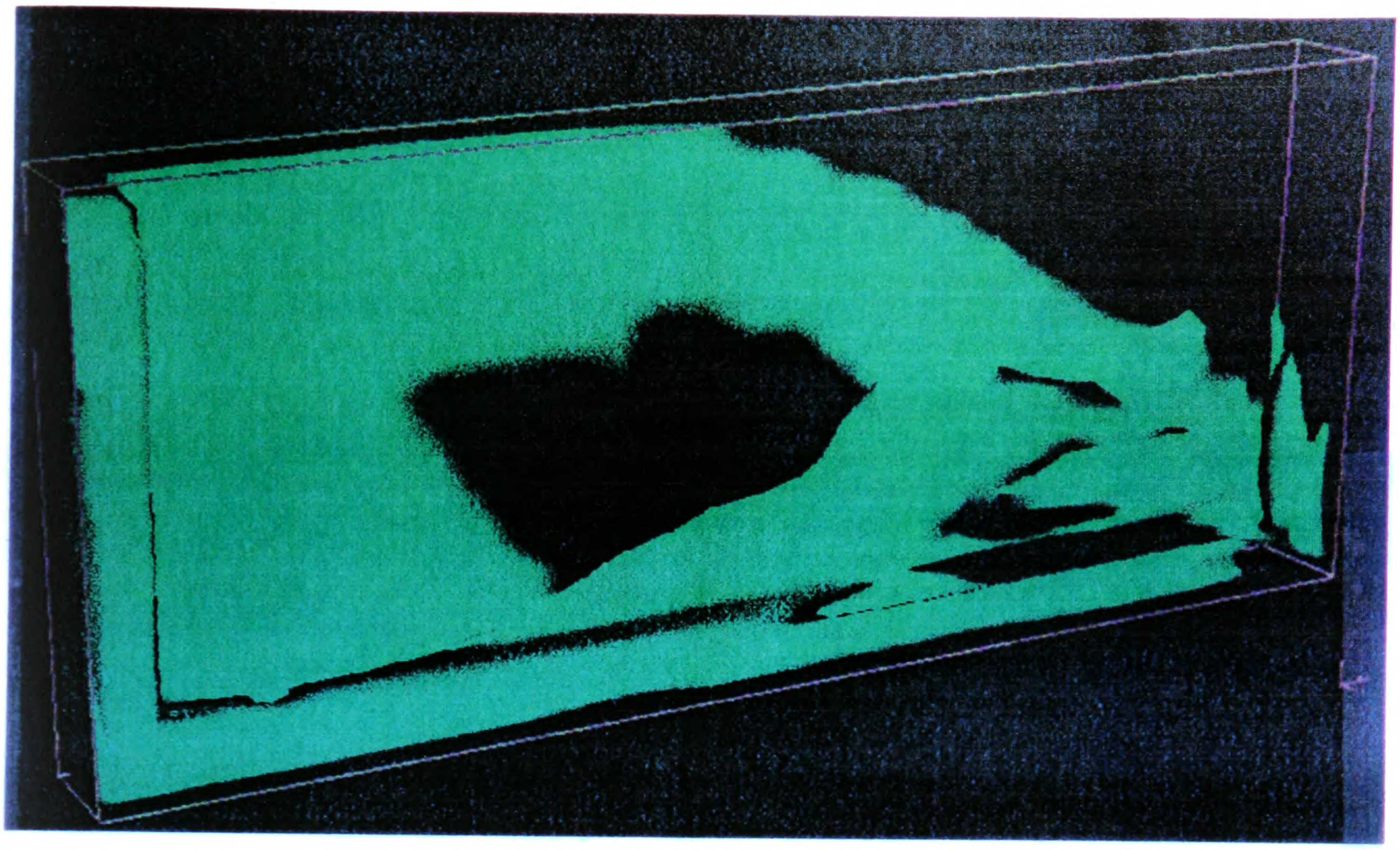


Figure 5.9 Visualisation of liquid free surface after 0.3 second.

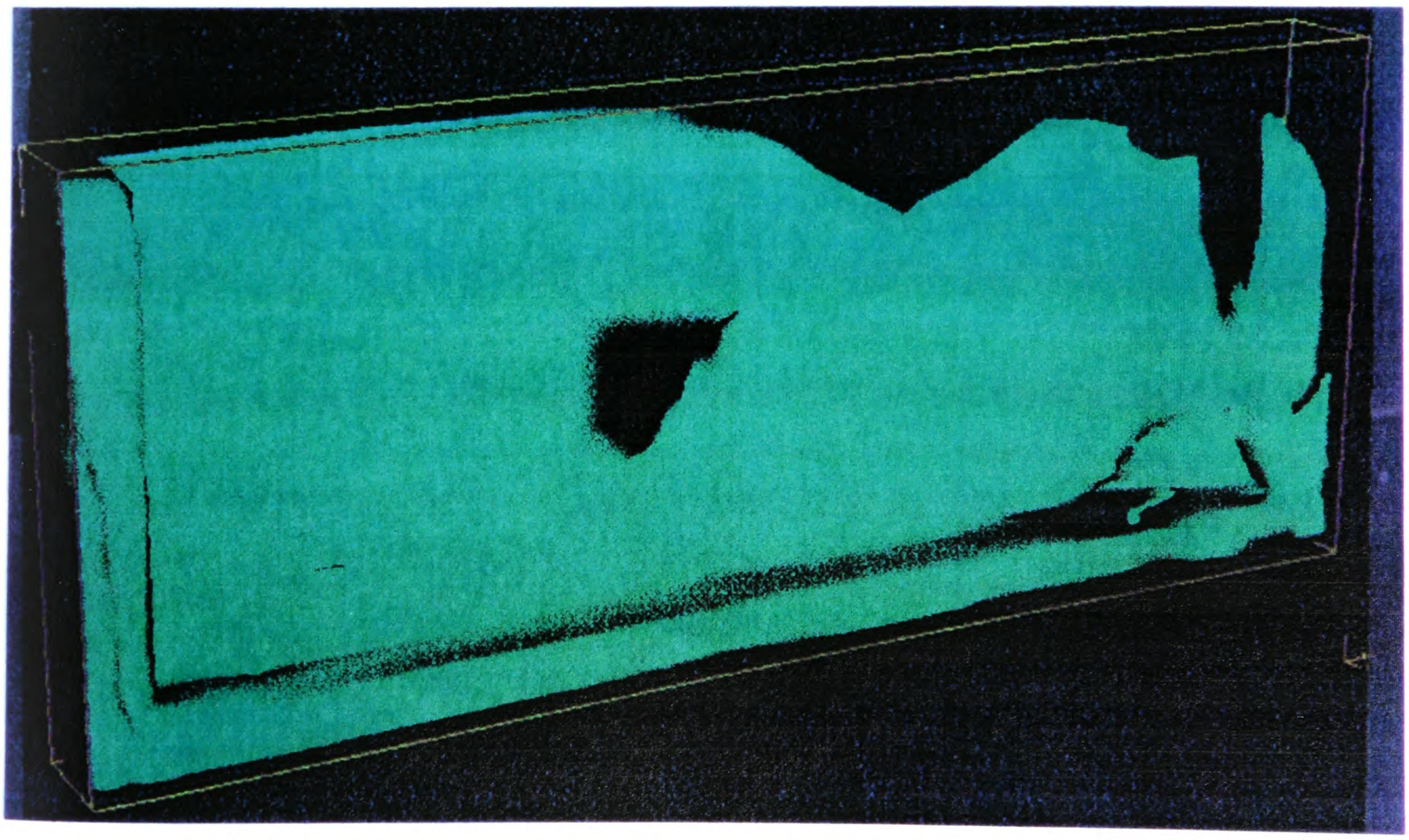


Figure 5.10 Visualisation of the liquid free surface after 0.4 second.

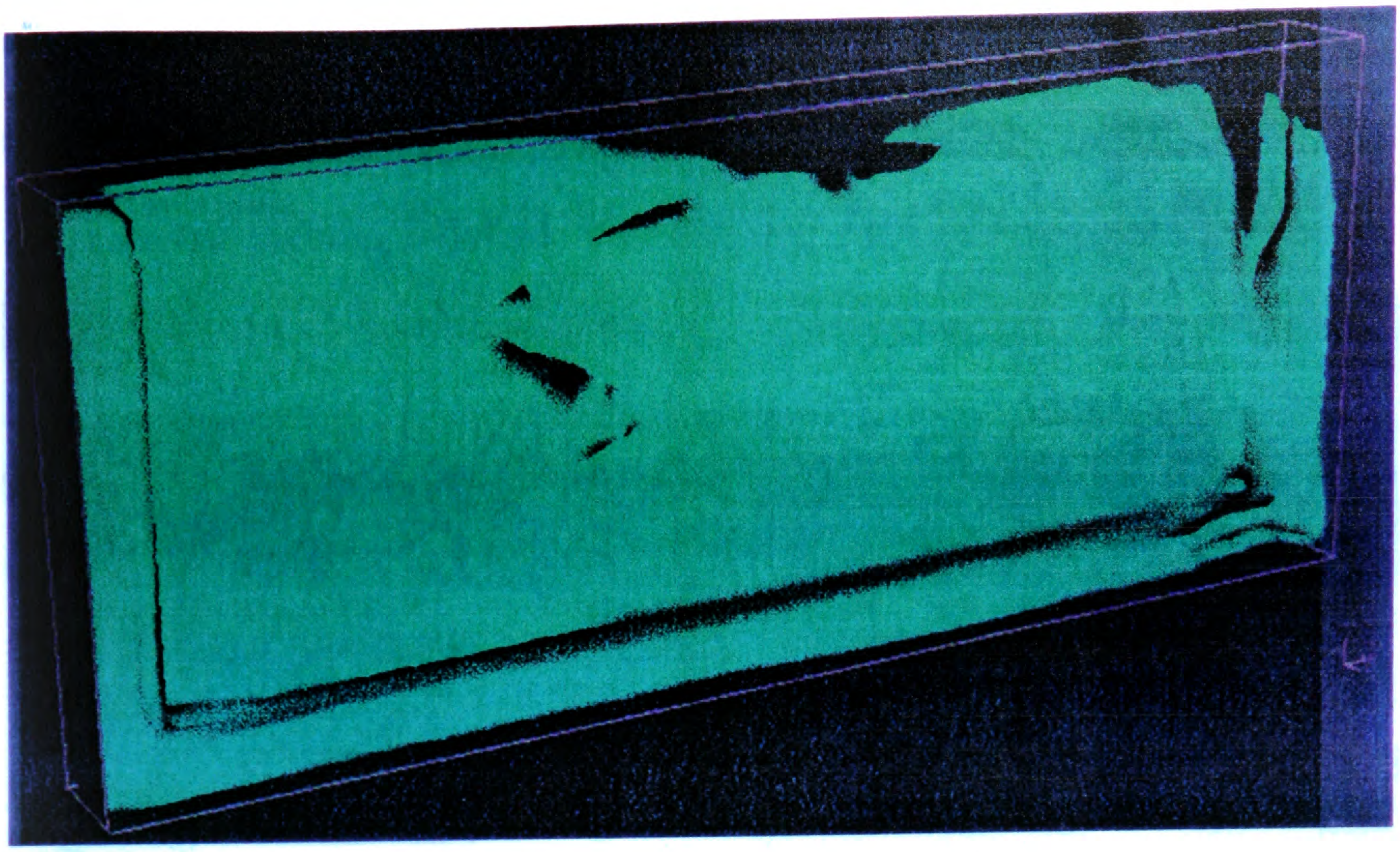


Figure 5.11 Visualisation of liquid free surface after 0.5 second.

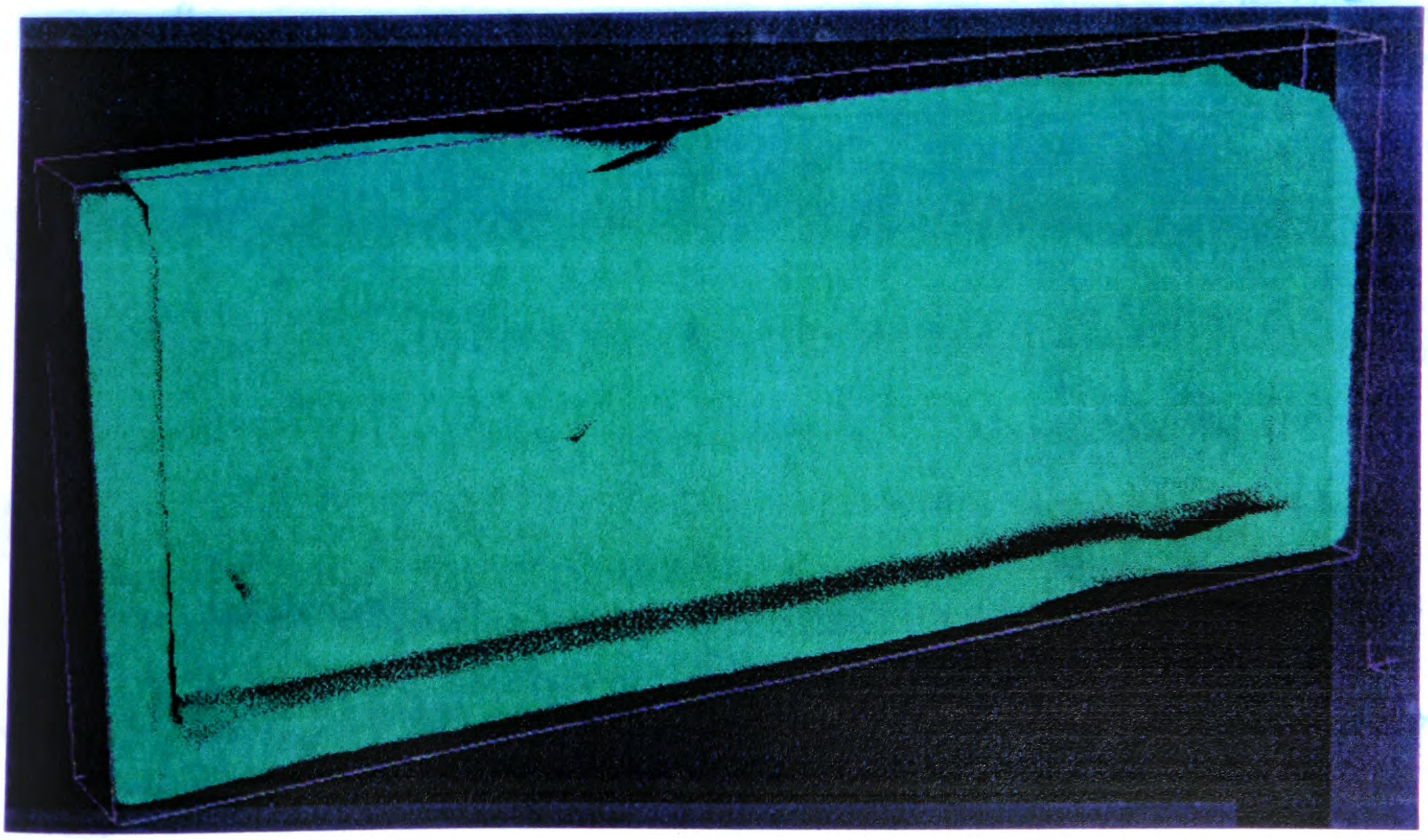


Figure 5.12 Visualisation of liquid free surface after 0.6 second.

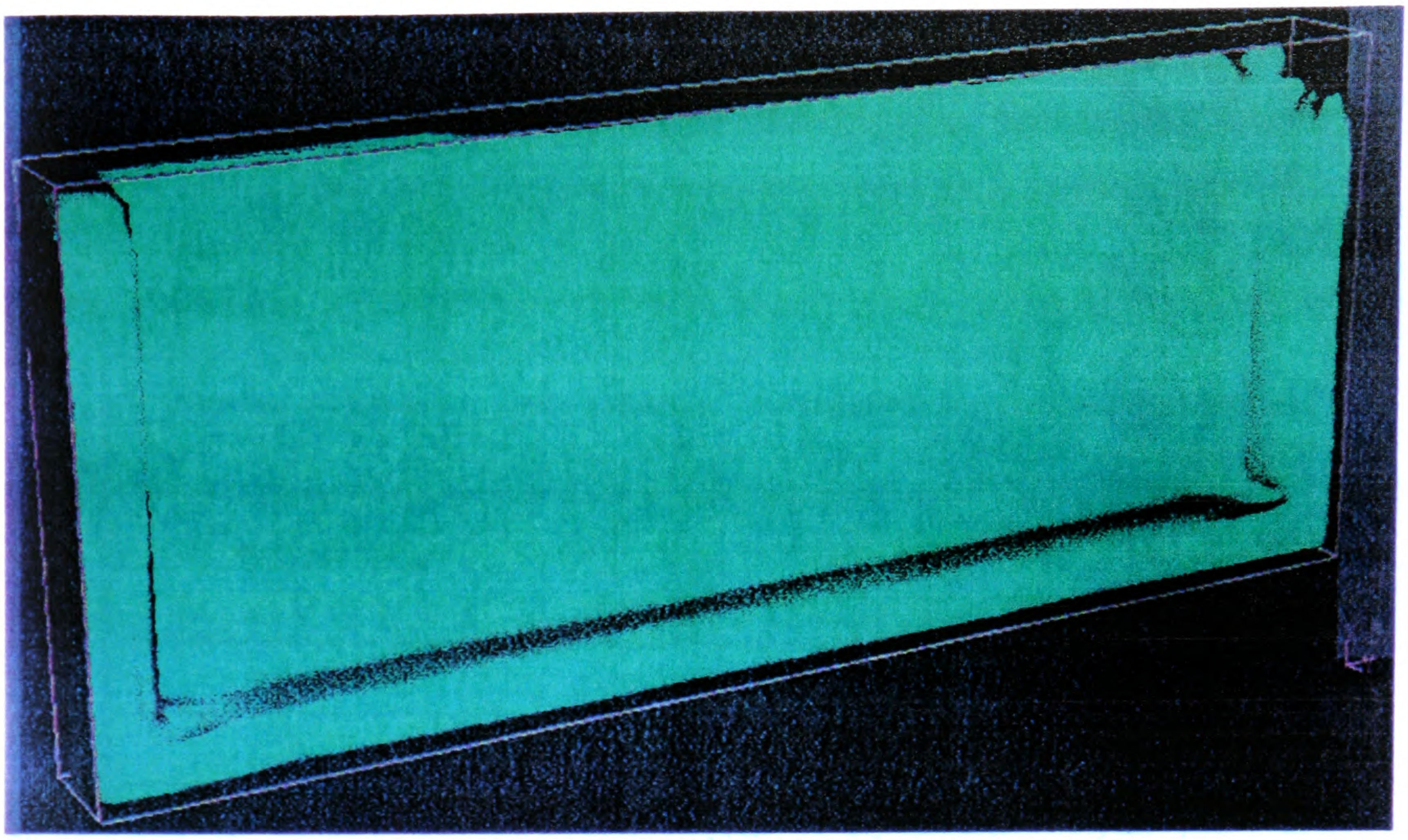


Figure 5.13 Visualisation of liquid free surface after 0.7 second.

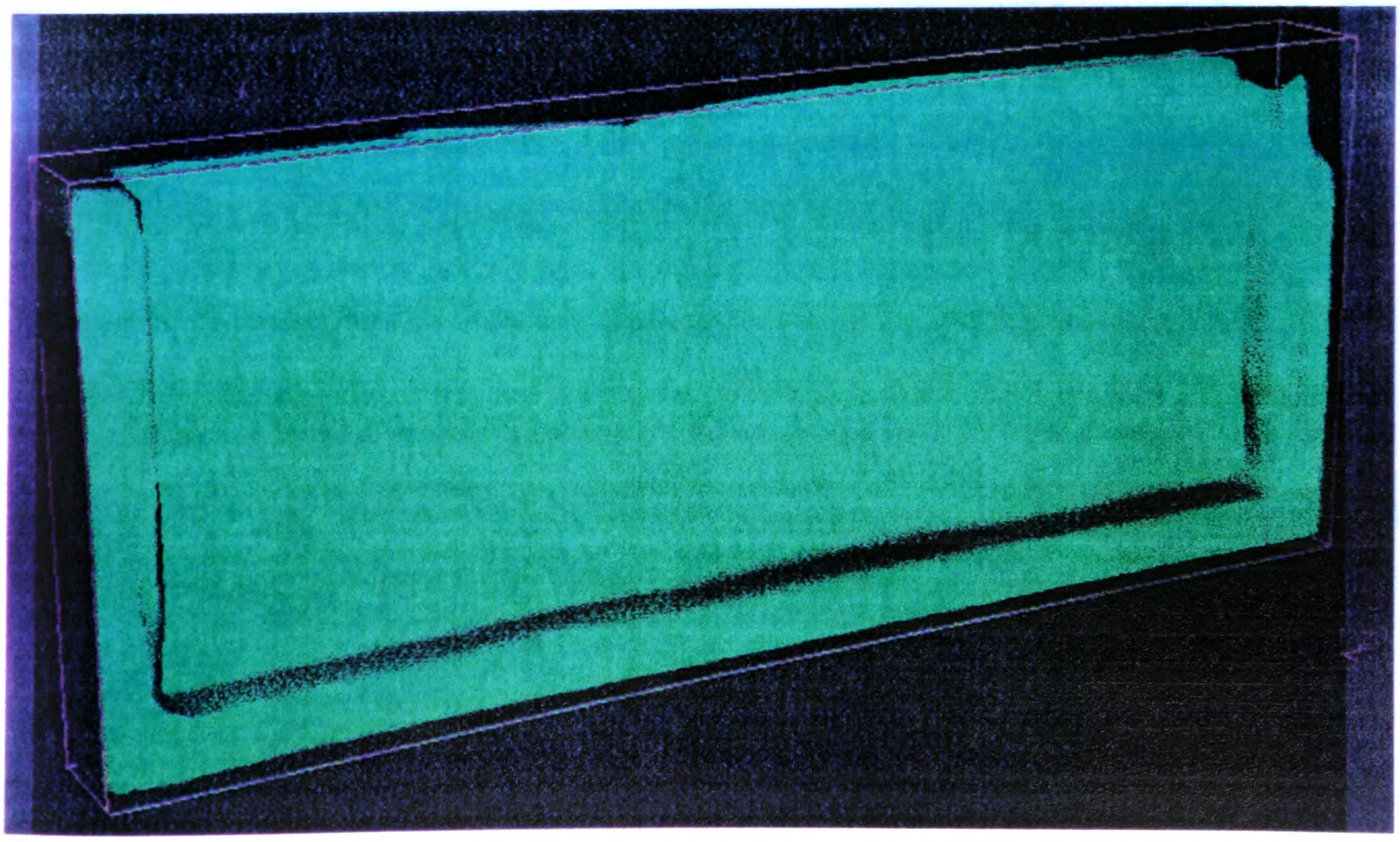


Figure 5.14 Visualisation of liquid free surface after 0.8 second.

Case II.

A finer mesh of 22 x 13 x 87 = 24882 cells with the same time step of 10^{-3} second and 800 time steps was used to carry out a series of runs on up to ten i860-transputer based parallel nodes. The overall CPU time of simulation running on ten i860s is about 10 hours. See *Table 5.3* for details.

Number of I860s	time (hours)	speed-up	Efficiency
1	51.4	-	-
2	28.5	1.8	90.0%
3	21.5	2.39	79.8%
4	17.8	2.88	72.0%
6	13.0	3.95	66.0%
8	10.96	4.69	58.6%
10	10.1	5.1	51.0%

Table 5.3 24882 cells, 0-0.8 seconds, running on i860-transputer system

Figures 5.15 to *5.17* depict the CPU time, speed-up and efficiency of the i860 hardware running with this mesh. With a much finer mesh of 22440 cells and a greater number of i860 nodes, a reduction in CPU time of 80% was achieved when comparing a single i860 node with ten i860s. Nevertheless, the similar trend of degradation in performance in both speed-up and efficiency of CPU utilisation can be observed from *Figures 5.16* and *5.17*. Except that this time, the degradation did not become apparent until four or more processors were used. There was no major gain in CPU time reduction by using ten i860s than using eight as the curve (of 24882 cells in *Figure 5.16*) is flattening out while the 'ideal' speed-up curve is linear.

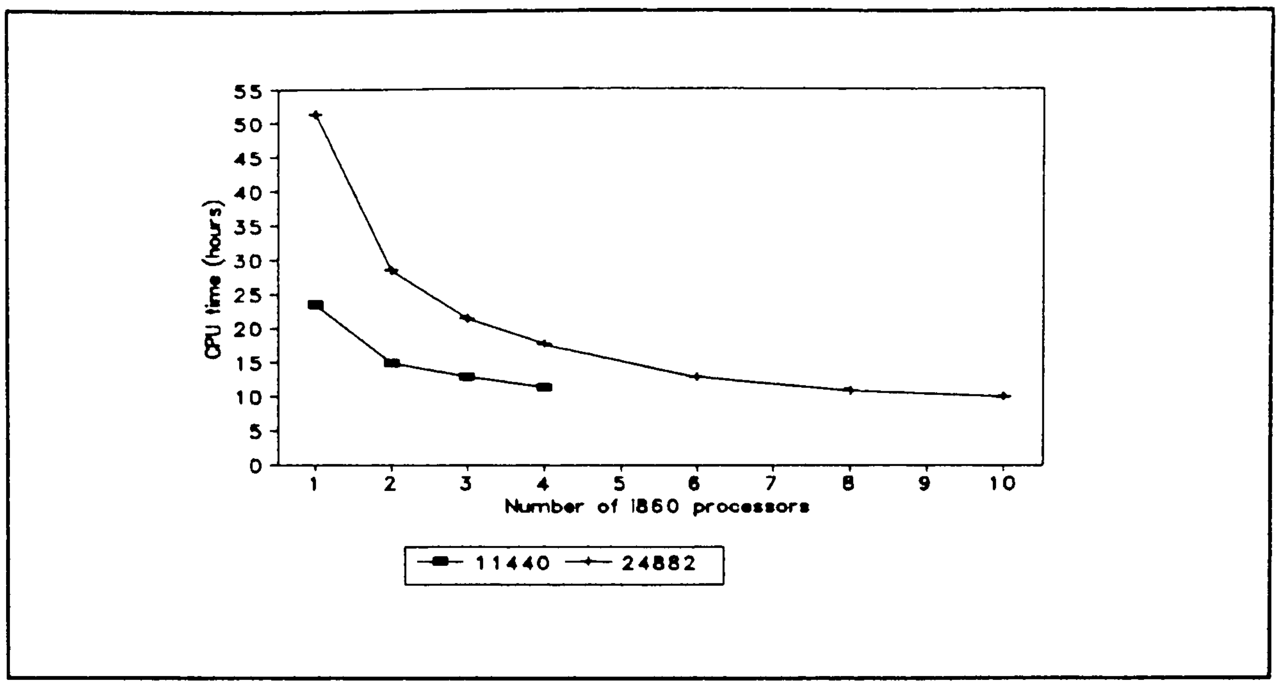


Figure 5.15 CPU time - i860 based transputer system

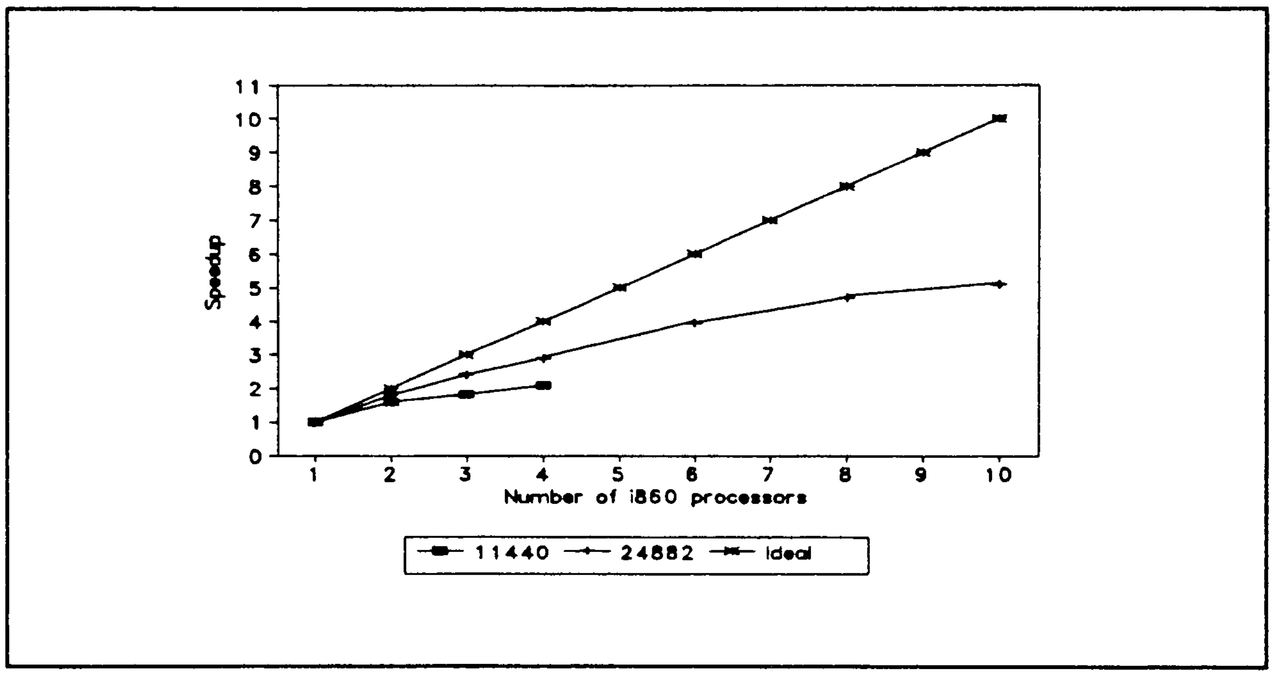


Figure 5.16 Speed-up - i860 based transputer system

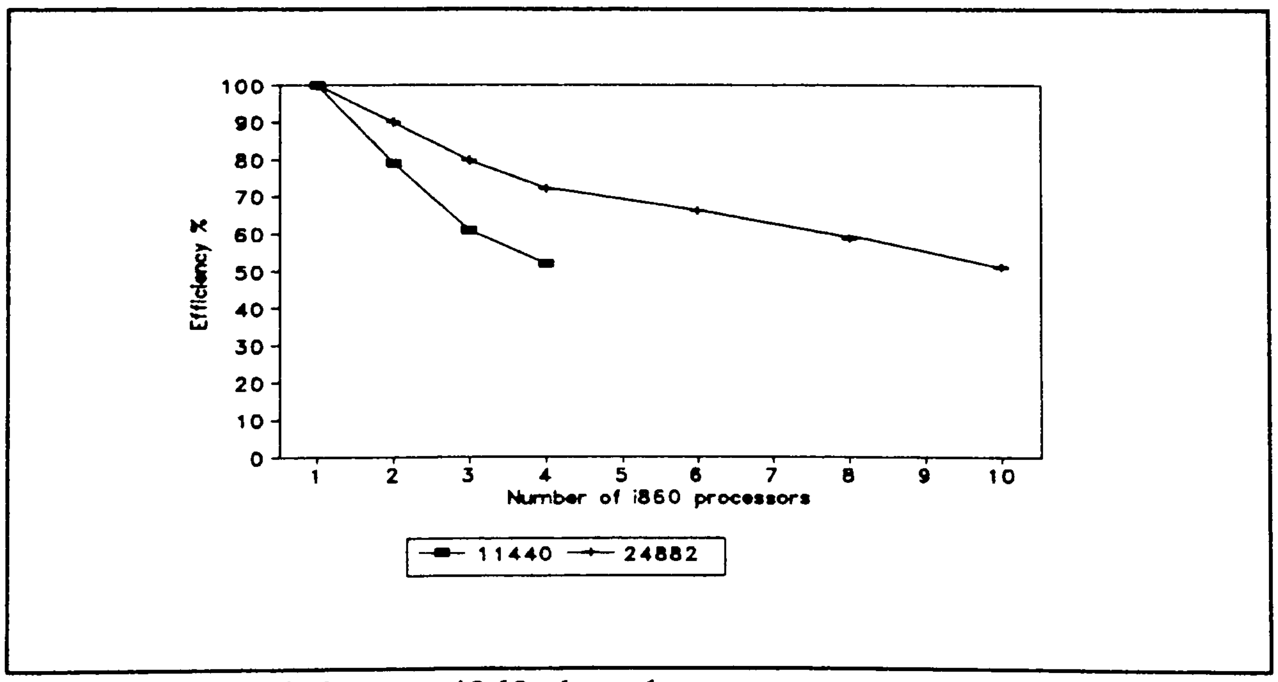


Figure 5.17 Efficiency - i860s based transputer system

Case III.

As for the transputer based parallel hardware, a coarse mesh of 11 x 10 x 17 = 1870 cells with a time step size of $5x10^{-3}$ seconds, and 21 iteration per time step was used on up to five T800s. The results of this simulation are very similar to that produced by the much finer grids using i860 processors. The overall simulation of 0.8 seconds in real time took 200 time steps to complete. When running on five transputers, it took about 1.27 hours to complete the simulation. See *Table 5.4* for details.

Number of T800s	time (hours)	speed-up	Efficiency
1	3.20	-	-
2	1.90	1.68	84.0%
3	1.60	1.97	65.8%
4	1.34	2.39	59.5%
5	1.27	2.53	50.7%

Table 5.4 1870 cells, 0-0.8 second, running on transputers only

Figures 5.18 to *5.20* depict the CPU time, speed-up and efficiency of the transputer runs. A reduction of CPU time from 3.2 hours on a single T800 to 1.29 hours on five T800s was achieved. The efficiency of transputer runs is similar in characteristic to those using the i860-transputer based hardware, however the order of magnitude is different.

5.6.3 Comments on the Parallel Results

The filling module implemented in the Harwell-FLOW3D code and running on the two hardware platforms, the transputer (T800) and the i860-transputer parallel architectures achieves similar efficiencies. While the number of cells or the computational load remains constant the efficiency decreases as the number of processors increases. The fall in the speed-up factor for cases I to III as the number of processors increased, may be due to the following factors:

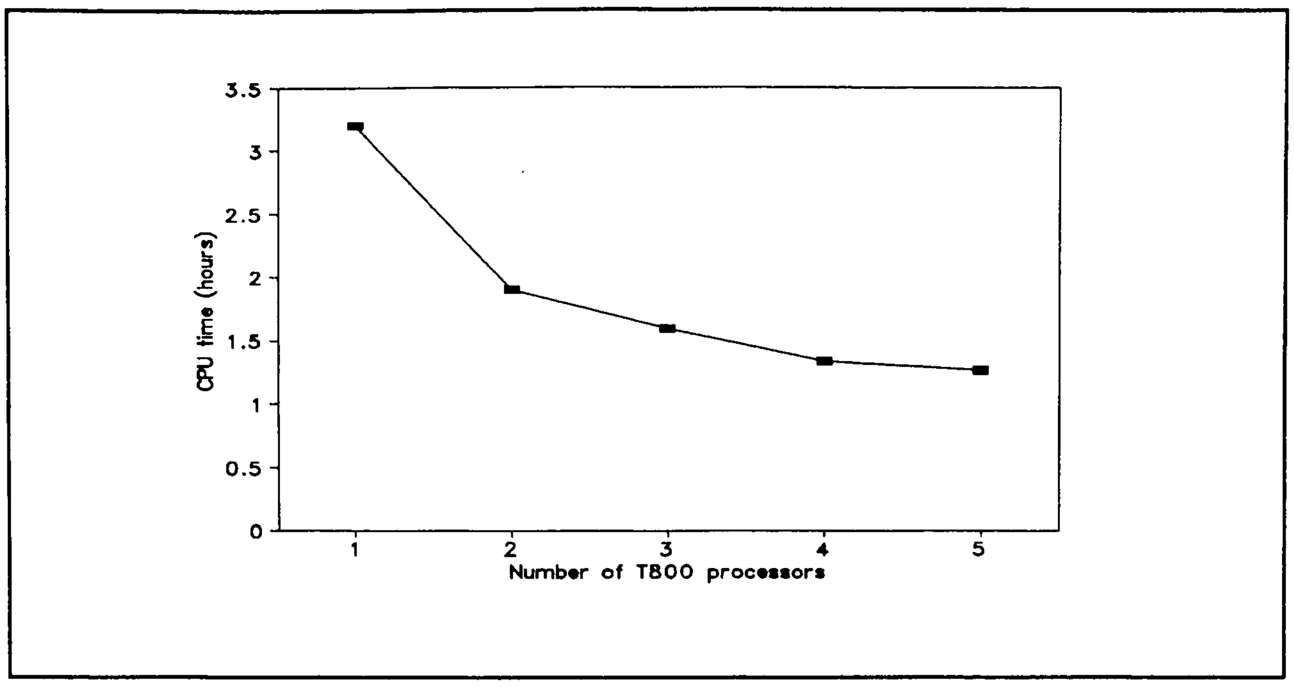


Figure 5.18 CPU time - T800 transputer system

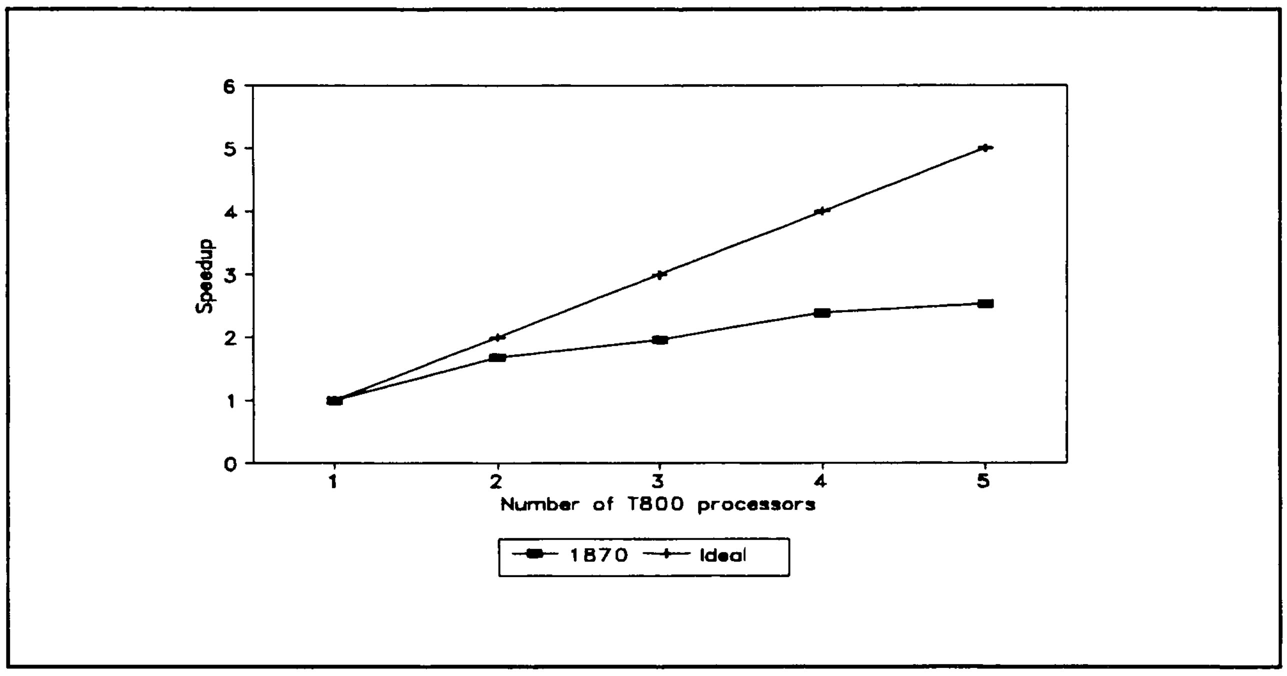


Figure 5.19 Speed-up - T800 transputer system

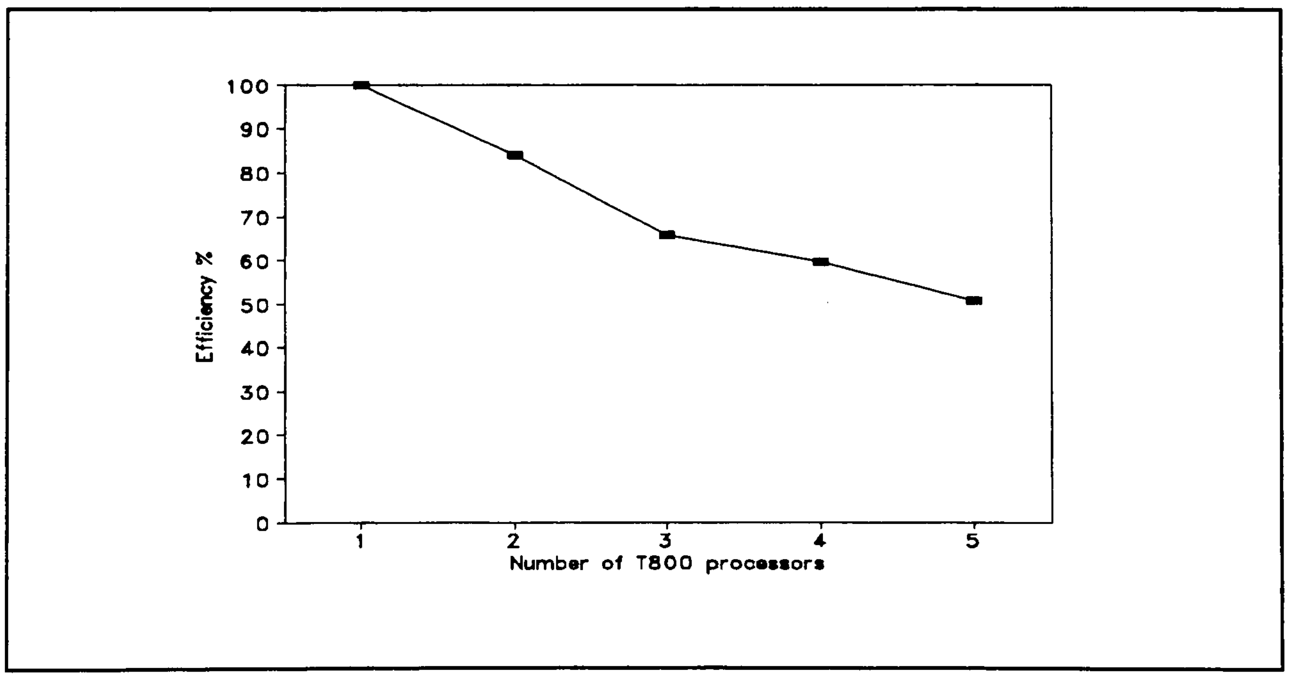


Figure 5.20 Efficiency - T800 transputer system

1. *Not enough computational load was assigned to each processor.*
   Since the communication process within each processor is running in parallel, expanding the number of processors does not increase the communication time. However, if the number of processors increases whilst the size of the computational domain remains the same, then the amount of data or computational load that can be assigned to each processor will be reduced. Therefore, the ratio of the calculation time over that of communication decreases, which causes the decline in efficiency.

2. *Due to load-imbalance when a different number of slabs were assigned to each processor.* For example, a low efficiency of 51% due to load-imbalance was obtained when running the finer mesh (24882 cells) with 87 slabs of data (in the z-direction) across 10 processors. This was caused as the master chip had to handle 15 slabs of data while the other slave chips only had to handle 8 slabs each.

3. *Due to latency in communication between the T805 transputer and i860.*

T805 and i860 are products of different era, in terms of processor design, complexity, and manufacturing technology. Therefore, they run at different clock speeds and hence have an inherent problem of communication latency. Based on tests carried out by S P Johnson [Johnson (1993)] and comparisons of running the container problem (using 1870 cells) on both a single T805 and i860 respectively; the i860 processor is found to be 10 to 14 times faster than a T805 in general. However, the ratio of start-up latency or delays in communication between the T805 and i860 is about one to four. Thus the communication over calculation ratio has degraded by a factor of 40 or more, obviously, this has a dramatic effect on the speed-up. This situation can be improved by having a more up-to-date and faster replacement for the T800, such as the T9000 transputer. If the communication link is running at a comparable speed to the i860's, then the ratio of latency may be reduced to unity. In order to achieve maximum efficiency the problem domain or computational load must be sufficiently large (subject to the maximum memory available) for each processor used in the network. In this way, a minimum amount of time is spent on communication alone.

5.7 Closure

The hypotheses set out in *Section 5.1.1* can now be answered.

(i) The filling module can be *ported* across to a CFD code generically, if the target code is structured and has a volumetric continuity procedure, such as GALA, built-in. For unstructured grid based CFD codes, such as ASTEC, the formulation for the current filling model will be quite different.

(ii) Running the filling model in parallel has shown that improvement in run-time efficiency can be obtained, especially when dealing with problem domains of 10,000 to 100,000 cells or more.

Reasonable speed-up and efficiency of processor utilisation can also be obtained provided that the issue of data balance (and/or computational balance) among processors is addressed properly.

Furthermore, it has been found by the author that it is very time-consuming and difficult to carry out research in the model while, at the same, maintain the parallel version of the code. Therefore, it would be useful if computer tools are available to help mathematical modellers to transform their codes from serial into parallel without spending too much time to understand the details of parallel programming. The Computer Aided Parallelisation Tools (CAPTools) is being developed at the University of Greenwich by S P Johnson and co-workers [Cross et al (1994)].

In general, the significance and possible impact that parallel computing can bring towards the scientific/engineering community by implementing the filling module into parallel Harwell-FLOW3D, has been described in this Chapter.

Chapter 6 *COUPLING FILLING WITH HEAT TRANSFER AND SOLIDIFICATION*

6.1 Introduction

The inclusion of heat transfer and subsequent solidification into the filling model is vital to the success of modelling mould filling numerically. Without the coupling of heat transfer and solidification with fluid flow, casting defects such as cold shut, blow holes, shrinkages and even oxide film formation, are not easy if not impossible to predict.

By and large, the results derived from simulation procedures, in which the filling, heat transfer and solidification processes are carried out one after another, are likely to be less realistic and may give misleading predictions. This is particularly true for the filling of short-range freezing alloys such as, the eutectic Aluminium-11% Silicon alloy (known as BS1490 or LM6), the Aluminium-Bronze alloy (BS1400 or AB2) and 0.2% Carbon steels. A simple explanation is that the transfer of thermal energy and the movement of fluid are taking place simultaneously in reality, therefore the inter-coupling between the two phenomena must also be simulated. For instance, the formation of a very thin shell at the metal liquid front will alter the flow velocity. Hence, the flow pattern could become very different to the case without heat transfer and solidification coupled. Also the mould will be heated non-uniformly by liquid jets.

To solve fluid flow, heat transfer, and solidification simultaneously under a single-phase framework is not a trivial task, as a multi-phase phenomenon (which involves gas, pure liquid, pure solid and even mushy zone) is simulated using a set of 'mixture' equations and volume fractions.

This Chapter is organised into the following sections:

- *Section 6.2* focuses on the development of a formulation to solve for heat transfer in a multi-fluid situation within a single phase framework.
- *Section 6.3* describes the formulation and implementation of a solidification model developed by Voller and Prakash (1987b) into the SEA algorithm coupled with heat transfer.
- *Section 6.4* describes an example of application using the SEA algorithm coupled only with heat transfer.
- *Section 6.5* describes an example using the SEA algorithm fully coupled

with heat transfer and solidification.

- *Section 6.6* gives a summary of this Chapter.

6.2 Incorporating Heat Transfer into the SEA Filling Model

The conservation of thermal energy which is characterised by the enthalpy equation (see *Equation 6.1* shown in x-direction only, or *Equation 2.4* in vector form, *Chapter 2*) cannot be used directly within the SEA filling algorithm without some modification.

$$\frac{\partial(\rho h)}{\partial t} + \frac{\partial(\rho u h)}{\partial x} = \frac{\partial}{\partial x}\left(\Gamma\frac{\partial h}{\partial x}\right) + S_h \qquad (6.1)$$

The problem is that the *enthalpy h,* as it stands, only represents the energy of a single fluid (either air or liquid) within the entire computational domain. When solving this equation with the SEA algorithm (described in *Chapter 3*), excessive numerical smearing for *h* exists around the air-liquid interface. This may be acceptable if the energy within the entire domain is conserved. However, the real problem occurs when evaluating the liquid temperature from the enthalpy *h*. Due to the severe numerical smearing the liquid temperature distribution is distorted to such an extent that the liquid thermal front is totally out of step with the air-liquid interface. The liquid temperature is essential in the modelling of solidification, due to the fact that the solidification process involves the absorption or evolution of latent heat energy (ΔH) which is always a function of liquid temperature [Voller and Prakash (1987b)]. However, the absorption or evolution of ΔH has no influence on the temperature of the material.

At first glance an 'easy' and obvious solution would be to solve heat transfer in a two-phase framework such as IPSA [Spalding (1977, 1980a)]. However, in the SEA algorithm when solving for the flow field the liquid and air are intentionally treated as a single fluid in volumetric form with the aid of the GALA algorithm [Spalding (1974, 1977)] . Therefore, it would seem to be irrational and wasteful to abandon the single-phase approach altogether and go for the two-phase calculations in order to solve for the energy equation. So a different approach is required to define and handle the *overall enthalpy* and subsequent *liquid temperature.*

6.2.1 Derivation of Mixture Enthalpy and Formulation of the Energy Equation

Consider the one-dimensional transient enthalpy equation (*Equation 6.1*) and using subscripts '*a*' and '*l*' to denote air and liquid phase respectively. Two equations, one for each fluid, can be derived as follows:

$$\frac{\partial(\rho h_a)}{\partial t} + \frac{\partial(\rho u h_a)}{\partial x} = \frac{\partial}{\partial x}\left(\Gamma_a \frac{\partial h_a}{\partial x}\right) + S_{h_a} \tag{6.2}$$

$$\frac{\partial(\rho h_l)}{\partial t} + \frac{\partial(\rho u h_l)}{\partial x} = \frac{\partial}{\partial x}\left(\Gamma_l \frac{\partial h_l}{\partial x}\right) + S_{h_l} \tag{6.3}$$

where $\Gamma = k/C_P$ is the thermal exchange coefficient which represents the ratio of conductivity over the specific heat for each fluid. The last term in each of the above equations represents other internal and external energy sources. These sources may include, for example,

(i) heat entering the domain due to bulk fluid motion such as the hot molten metallic liquid entering the mould cavity;

(ii) heat transfer between sand walls and the fluid; and

(iii) heat transfer at the interface between the two fluids (ie. air and liquid).

In addition, if solidification or change of phase of liquid into solid is taken into account, then the source term will contain the contribution from the latent heat.

By definition of *Equation 3.6* or *3.7* (in *Chapter 3*), the properties of the 'mixed' or overall fluid in the domain of interest are defined by two volume fractions: ϕ for liquid and $(1 - \phi)$ for air. *Equations 6.2* and *6.3* are integrated over a typical control volume (see *Figure 2.4*, *Chapter 2*), and multiplied by the appropriate volume fraction. By summing the two equations the 'mixture' energy equation in finite difference form for a one dimensional model in x-direction is then given as:

$$\begin{aligned}
&\frac{V}{\delta t}\Big[(\rho_l\phi h_l + \rho_a(1-\phi)h_a) - (\rho_l\phi h_l + \rho_a(1-\phi)h_a)^{old} \Big] \\
&+ \Big[(\rho_l\phi h_l + \rho_a(1-\phi)h_a)_e u_e A_e - (\rho_l\phi h_l + \rho_a(1-\phi)h_a)_w u_w A_w \Big] \\
&= \left[\phi\Gamma_l\frac{\partial h_l}{\partial x} + (1-\phi)\Gamma_a\frac{\partial h_a}{\partial x} \right]_e A_e - \left[\phi\Gamma_l\frac{\partial h_l}{\partial x} + (1-\phi)\Gamma_a\frac{\partial h_a}{\partial x} \right]_w A_w \\
&+ \Big[\phi S_{h_l} + (1-\phi)S_{h_a} \Big] V
\end{aligned} \tag{6.4}$$

where V is the cell volume, A the cell face area, the superscript old refers to 'old time' values and subscripts e and w refer to the east and west cell faces (see *Figure 2.4*). A new variable referred to as *mass averaged enthalpy* can be defined as follows:

$$\bar{h} = \frac{\rho_l\phi h_l + \rho_a(1-\phi)h_a}{\rho_l\phi + \rho_a(1-\phi)} \tag{6.5}$$

where the denominator is essentially the average volumetric density in the cell, ρ, and the numerator is simply the average specific heat content of the cell.

Equation 6.4 can then be simplified further by this new variable and can be expressed in terms of average enthalpy. Furthermore the diffusion gradients can be replaced by their finite-difference equivalents to give,

$$\begin{aligned}
&\frac{V}{\delta t}\Big[(\rho\bar{h})_P - (\rho\bar{h})_P^{old}\Big] + \Big[(\rho\bar{h})_e u_e A_e - (\rho\bar{h})_w u_w A_w\Big] \\
&= \left[\left(\frac{k}{C_p}\right)_e (\bar{h}_E - \bar{h}_P)\frac{A_e}{(x_E - x_P)} + \left(\frac{k}{C_p}\right)_w (\bar{h}_P - \bar{h}_W)\frac{A_w}{(x_P - x_W)}\right] \\
&+ \Big[\phi S_{h_l} + (1 - \phi)S_{h_a}\Big]
\end{aligned} \tag{6.6}$$

where the subscripts P, E and W are used to denote the enthalpy in the central node P and the east and west node neighbours (see *Figure 2.4*, *Chapter 2*). The lower subscripts refer to cell face quantities. The above equation reverts back to *Equation 6.2* and 6.3 when the volume fraction ϕ has value of either 0 or 1; ie. away from the air-liquid interface region.

Equation 6.6 is arranged in three dimensions (ie. in x, y and z directions) in a usual way as described in *Chapter 2*. Once all the convective and diffusive fluxes have been calculated at each cell face, the new value of h_p is obtained by solving *Equation 6.6* with one of the TDMA type solvers available in PHOENICS. For staggered grid, the material properties, enthalpy h and other scalar variables are stored at the cell centre while the velocity components are stored at cell faces as described in *Section 2.3.2.1* of *Chapter 2*. Appropriate differencing schemes are therefore required to approximate the value of these scalar variables at the cell faces. In order to prevent numerical smearing of liquid enthalpy across the air-liquid interface, the convection and diffusion terms use the value of ϕ at the cell face derived using the van Leer scheme [van Leer (1977)] in the mixture property formulae (*Equations 3.5* and *3.6* of *Chapter 3*).

6.2.2 Effect of Heat Transfer to the Momentum Equation

An external source term is needed in the conservation of momentum equation (*Equation 2.3*, *Chapter 2*) to account for natural buoyancy effect due to thermal expansion. This term is only applicable to the vertical velocity component.

• Buoyancy Term

The buoyancy term S_B for a material with constant property can be estimated by the Boussinesq approximation as given below:

$$S_B = -\rho_{ref}\beta g(T - T_{ref}) \tag{6.7}$$

where ρ_{ref} is the *reference density* and it is usually set to the constant density of liquid ρ_l. T_{ref} is the *reference temperature* which is usually set to room temperature, g is the gravitational acceleration in the vertical direction, and β is a constant known as the volumetric thermal expansion coefficient.

However, when modelling heat transfer with filling, the buoyancy term becomes less significant and may be ignored, as the bulk of the fluid is moving under the influence of the inflow velocity. In other words, the flow is under forced convection.

6.2.3 Determination of Liquid Temperature

With the assumption of constant specific heat of a fluid [Voller and Prakash (1987b)], it is possible to derive the temperature T from the enthalpy h by the equation below:

$$h = C_p T \tag{6.8}$$

Therefore it is feasible to derive the liquid temperature (*Equation 6.9*) from a fluid mixture at the interface region by using the relationship stated in *Equation 6.5*, if certain assumptions have been made.

6.2.3.1 Assumptions for Liquid Temperature

1. *In the case of liquid-air mixture, the air enthalpy contribution is negligible for* $\phi > 0.1$*, say.* This is due to the large density ratio between liquid and air which is over a factor of 1000.

2. *The air temperature is assumed to be constant and known in advance.* This is reasonable and necessary since it is the liquid temperature which is of most interest. Furthermore, mathematically it is not possible to solve a single equation with two unknowns.

The temperature for liquid metal is defined as follows:

$$T_l = \frac{\rho \bar{h} - (1-\phi)\rho_a h_a}{\phi \rho_l C_{P_l}} \tag{6.9}$$

For other situations where the density difference is not so great, it is reasonable to assume that at a suitable volume fraction limit, $\phi_{\lim}$ exists. Then the minority fluid takes the temperature of the dominant fluid which surrounds it. Then the following formula is more appropriate:

$$\begin{aligned} &\textit{If } \phi \geq \phi_{\lim}, \quad T_a = T_l \Rightarrow T_l = \mathcal{T} \\ &\textit{If } \phi < \phi_{\lim}, \quad T_l = T_a \Rightarrow T_a = \mathcal{T} \\ &\textit{where } \mathcal{T} = \frac{\rho \bar{h}}{\phi \rho_l C_{P_l} + (1-\phi)\rho_a C_{P_a}} \end{aligned} \tag{6.10}$$

Equation 6.9 has been used in the simulations presented in *Sections 6.4* and *6.5* later. The liquid temperature was found to be insensitive to a ϕ_{lim} value as high as 0.5. Furthermore, the sharpness of the liquid temperature contour at the gas-liquid interface is a dependent of the value of the metal fraction ϕ which in turn depends on the accuracy of the van Leer scheme used.

6.3 Coupling of Solidification with Heat Transfer and Fluid Flow

A number of control volume based solidification models are available for modelling the phenomenon of phase change within a material (see *Section 1.4*, *Chapter 1*). Among them the model developed by Voller and Prakash (1987b) is chosen for this work for reasons as explained in the following section.

6.3.1 The Solidification Model

In the case of pure solidification, the total heat content (H) of a material can be defined as follows:

$$H = h + \Delta H \tag{6.11}$$

where h, in this case, is known as the *sensible heat* and ΔH is the *latent heat* energy as defined earlier. The solidification process relies on the evolution or absorption of the latent heat energy (ΔH) [Crank (1984)]. As mentioned in *Section 6.2*, ΔH is a function of the temperature of the material, that is ΔH = f(T). But the function f(T) depends on f_l and L which are the *liquid fraction* and the *latent heat of phase change* respectively. Hence, ΔH can also be represented by f_lL.

In the case of filling with solidification, the ΔH for liquid to solidify can be treated as a source term in the energy equation (*Equation 6.1* or *6.3*). The source term (or sink in this case) consists of transient and convective parts, as shown below:

$$\overline{S}_{h_l} = -\frac{\partial(\rho f_l L)}{\partial t} - \nabla.(\rho \underline{u} f_l L) \tag{6.12}$$

where f_l represents the proportion of metallic mass which remains in the liquid state. This quantity should not be confused with ϕ, which is the volume fraction of liquid used by the SEA algorithm during filling. In order to avoid any confusion in the later part of this thesis, ϕ now refers to the '*metal fraction*'. The definition of the liquid fraction f_l is given in *Equation 6.13* below:

$$f_l = \begin{cases} 1 & \textit{Liquid for } T > T_L \\ \left(\dfrac{T - T_S}{T_L - T_S}\right) & \textit{Mushy Zone for } T_S \leq T \leq T_L \\ 0 & \textit{Solid for } T < T_S \end{cases} \tag{6.13}$$

where T_L is the *liquidus* temperature and T_S is the *solidus* temperature.

Primarily, f_l is a step function of the temperature of the material. When the temperature of the material is above its liquidus temperature, the material remains in the liquid state and hence f_l is one. As thermal energy is dissipated via convection, conduction or even radiation through the mould walls and the air inside as well as outside the mould cavity, the temperature of the metallic liquid within the mould cavity always decreases. When metallic liquid transforms itself into solid state, it undergoes an intermediate phase of the so-called *mushy zone*, in which partially solidified metal, in the form of dendrites, is suspended in the sea of jelly-like molten liquid. This intermediate phase only occurs if the material is not of eutectic composition or a pure metal [Polmear (1991)] and when the temperature of the liquid is between its *liquidus* and *solidus* temperatures. The solid state of the material is reached when the temperature drops below its solidus point. Furthermore, a number of alternative mathematical expressions to model the mushy region are available, see the review by Voller and Swaminathan (1991). A linear one adopted by Voller and Prakash (1987b) has been chosen for this study for its simplicity and robustness. Also, it does not require any relaxation adjustment.

The liquid fraction f_l for each cell is updated at every iteration step from m to $m+1$ as revealed in *Equations 6.14* and *6.15* below:

$$f_l^{m+1} = f_l^m + \Delta \tag{6.14}$$

where Δ is the *liquid fraction correction* term, and which is defined as,

$$\Delta = \frac{C_{P_l}\left[T_P^{m+1} - [(T_L - T_S) f_L^m + T_S]\right]}{L + (T_L - T_S)\, C_{P_l}} \tag{6.15}$$

The liquid fraction f_l is converged when Δ approaches zero or a very small prescribed tolerance.

6.3.2 Effect of Solidification to the Momentum Equation

In order to model solidification correctly, an external source term is needed in the momentum equations (*Equation 2.3*, *Chapter 2*). The first one is the so-called Darcy term which applies to all three velocity components.

• Darcy Term

The formation of dendrites and other crystalline forms during solidification creates a mushy zone, in which the velocity of liquid metal is slowed down. Following other researchers [Voller et al (1985, 1987b, 1991) and Chow (1993)] , a Darcy resistance term is introduced in the momentum equations (*Equation 2.3*, *Chapter 2*), as an external source/sink to suppress the velocity components as the material undergoes solidification. These sink terms are shown in the equations below:

$$S_u = -\frac{\mu}{K} u$$

$$S_v = -\frac{\mu}{K} v \tag{6.16 a to c}$$

$$S_w = -\frac{\mu}{K} w$$

where K represents the permeability, given by the Carman-Kozeny equation as below:

$$K = \frac{f_l^3}{C(1 - f_l)^2} \tag{6.17}$$

In the above equation, C is a large positive constant which depends on the morphology of the porous medium [Voller and Prakash (1987b)]. The value of C in the case of Voller and Prakash (1987b) was taken as $1.6\text{x}10^5$. The value of C varies depending on the molecular structure of the material undergoing solidification. For instance, in the case of Samonds and Waite (1993), an expression of $C = 5M_s^2$ was used. In which M_s is the volumetric specific surface area of a typical dendrite arm and it is approximated by assuming that the shape of a dendrite arm is conical.

6.3.3 Solution Procedure for Fluid Flow, Heat Transfer and Solidification

The dependent variables to be solved in a full-scale coupling of fluid flow, heat transfer and solidification phenomena are: velocity components (u,v,w), pressure (p), and mixture enthalpy (h). Other auxiliary variables which need to be updated are: '*metal fraction*' (ϕ) which is explicitly calculated by a van Leer scheme, liquid temperature (T_l) which is determined from the mixture enthalpy, and the liquid fraction (f_l) which is derived from the liquid temperature as explained above.

The partial differential equations for momentum and thermal energy are converted into finite difference equations by integrating over each CV cell in the computational domain as explained in *Chapter 2*. Consequently, a set of algebraic equations is formed for each variable in a particular cell and its immediate neighbours. The hybrid interpolation difference scheme [Spalding (1972)] is used to evaluate convective and diffusive fluxes at the cell faces, except for the scalar advection equation (*Equation 3.5*, *Chapter 3*) which instead uses the van Leer scheme. In addition, the cell face fluxes (other than those of ϕ) are corrected by the appropriate face values of ϕ via the density and viscosity.

The resulting set of simultaneous finite difference equations are then solved iteratively by PHOENICS using the built-in SIMPLEST pressure correction algorithm. The full solution procedure steps are listed below:

1. Solve momentum equations using a guessed pressure field.
2. Compute volumetric continuity error (due to the use of GALA) and hence form a pressure-correction equation. Solve using a slab-by-slab or whole-field three-dimensional solver.
3. Apply corrections to pressure and velocities to ensure continuity is satisfied.
4. Repeat until continuity errors fall below predetermined limits.

5. Update liquid fraction f_l using the liquid temperature available and hence determine the amount of latent heat (ΔH) released during solidification.
6. Solve enthalpy and hence determine the latest liquid temperature.
7. Update the scalar field ϕ using the latest velocities with van Leer scheme.
8. Update corresponding fluid properties and repeat steps 1 to 7.
9. On convergence proceed to next time step.

Steps 5 to 7 have been implemented by the author.

6.4 Example I: Filling with Heat Transfer only - Step Wedge

The geometry of the step wedge (described in *Section 4.5*, *Chapter 4*) is used to demonstrate the characteristics of the heat transfer technique introduced in *Section 6.2*. The effect of phase change or solidification was not simulated in this case. An example of coupling free surface flow, heat transfer and solidification can be found in *Section 6.5*.

The way the liquid surface behaves is of particular interest in this example as waves may lead to the breaking of the metal surface resulting in the entrainment of oxides and other impurities into the casting. This is also known as the formation of an oxide film. The air entrained in the liquid must be allowed to escape before the liquid metal solidifies, otherwise 'blow holes' or gas porosities may form in the casting. This kind of defect is most common in high pressure-die castings [Campbell (1991)].

The temperature distribution in the liquid immediately after the mould has been filled is of most interest. It affects the subsequent solidification process as mentioned in *Section 6.1*. Careful control of heat transfer, and hence the temperature gradient through the body of the sand mould, is vital in making sound castings free of shrinkage and porosity formation in both macro and micro scales. The removal of waves and air entrainment can be performed by careful design of the running system, ie. by positioning the ingate(s) and feeders at appropriate locations such that the metallic liquid filling into the mould is not turbulent [AFS Cast Metals Technology Series (1976a)]. In that way smooth and steady flow of liquid metal is filling the mould avoiding the formation of waves. It is these waves and their subsequent surface breaking which would cause air and/or impurity entrainments. To induce the correct solidification direction for the liquid metal, the appropriate temperature gradients for cooling can be achieved by the use of 'metal chills' at strategic locations. The use of 'ceramic sleeves' is to maintain the liquid in the feeders at a higher temperature than those inside the mould. Most importantly, placing the feeders on top of 'cold spots' can eliminate 'cold-shut' or 'mis-feed' and shrinkages [AFS Cast Metals Technology Series (1976b)].

6.4.1 Problem Specification

A mesh of 20 x 28 = 560 cells, identical to the one used in *Section 4.5.1*, was used to carry out the simulation in two-dimensions. The following material properties (*Table 6.1*) have been assumed for the two fluids and the sand mould in this simulation:

Materials	Aluminium Alloy	Air	Sand
Density ρ, (kg/m^3)	2800.0	1.2	2000
Kinematic Viscosity ν,(m$^{2/s}$)	3.6x10^{-7}	2x10^{-5}	-
Specific Heat C_p, (J/kg°K)	880.0	993	676
Thermal Conductivity k, (W/m°K)	180.0	0.024	0.753

Table 6.1 Material properties for step mould.

The initial temperature of the mould and air was taken to be 300°K (which is about the usual temperature found in a foundry environment), and the liquid metal is assumed to enter at a temperature of 1300°K. Liquid enters the mould through a gate (see *Figure 6.1a*) at a constant velocity of 0.25 m/s. Air exits the mould at the top of the feeder or riser section where the pressure is assumed to be atmospheric. Although in reality the sand mould is porous and hence permeable, in the present simulation air is prevented from escaping through the sand. To further simplify the calculation the temperature of the mould is assumed to remain constant during the filling process. This exaggerates the cooling effect of the walls and leads to temperatures which lie below the solidus value 913°K. However, since solidification was not included in the calculation, this is of no physical significance. The full simulation in two-dimensions took approximately one CPU-hour to complete on a Sun Sparc IPX workstation.

6.4.2 Results and Remarks

The results of the simulation are given in *Figures 6.1* and *6.2* which follow the filling process from 0.35 to 2.5 seconds, by which time the feeder is already full of liquid. Following the convection of the results presented throughout *Chapter 4*, plots are given of the vector field and the free surface contour (at $\phi = 0.5$). *Figure 6.1* shows the distribution of free surface interface at the four time intervals (0.35 second, 0.5 second, 1 second, and 2.5 seconds), and *Figure 6.2* shows the corresponding temperature distributions.

Figure 6.1a shows the liquid layer tapering as it accelerates under the action of gravity. As the liquid penetrates into the tapering section of the mould, the air which is initially stagnant is expelled thus forming a counter-current stream above the liquid surface. The resulting shear layer causes the liquid surface to swell, so that contact is made with the mould top surface before the air has been completely expelled. As a consequence, a series of air pockets are formed which can be seen in *Figures 6.1b* and *6.1c*. Most of this air is subsequently expelled, until at 2.5 seconds (*Figure 6.1d*) only a small layer remains at the narrowest point of the casting. The velocity vectors indicate that at his stage there is very little flow in the far end of the thinnest section of the mould. The incoming fluid flow, is instead diverted upwards along the far wall of the feeder.

Figures 6.2a to *6.2d* show the corresponding temperature distribution. The reader is reminded that the cooling rate, in this example, has been increased for illustration purposes. Initially the temperature of liquid is uniform everywhere, except close to the floor of the mould (*Figure 6.2a*). At 0.75 second (*Figure 6.2b*) substantial cooling

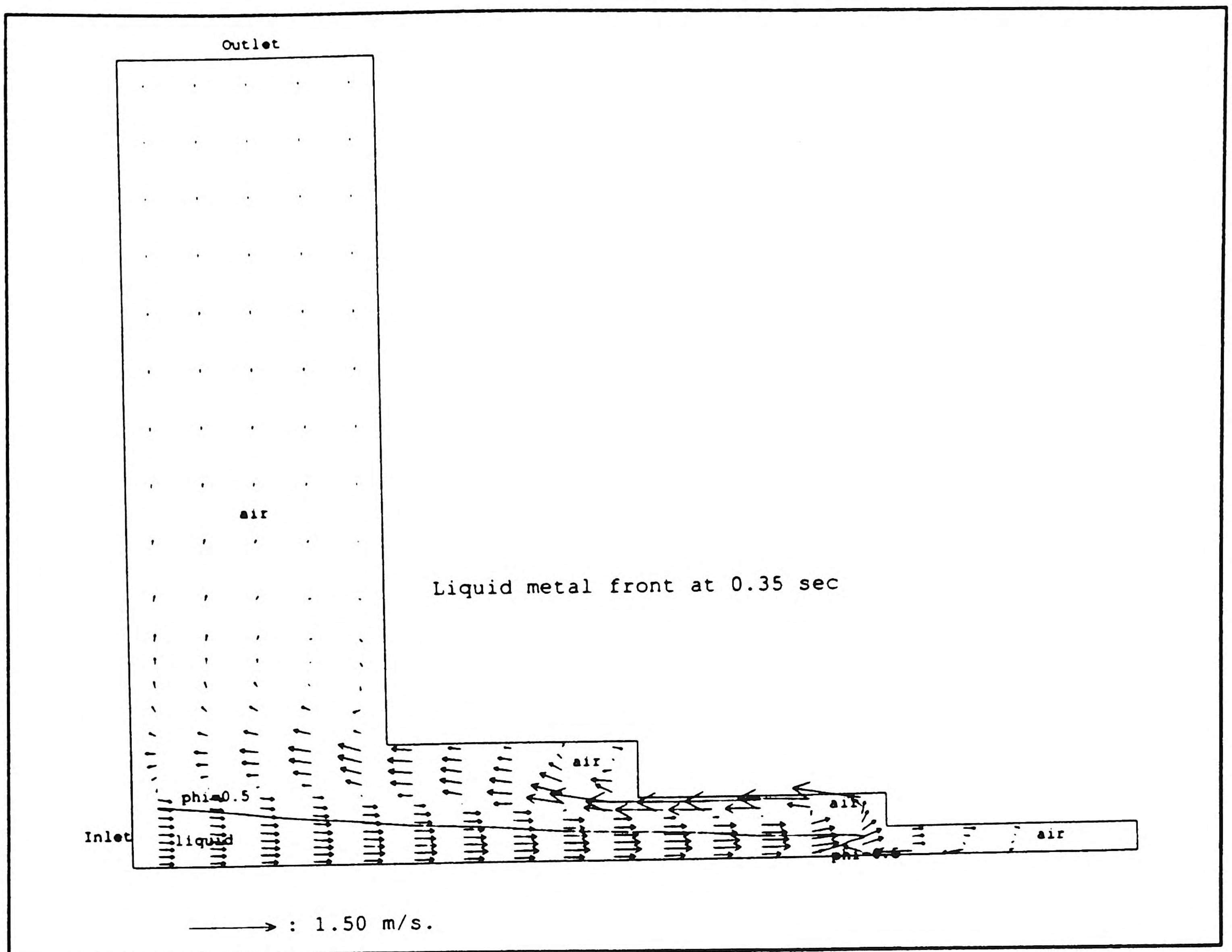


Figure 6.1a Velocity vectors and free surface after 0.35 second.

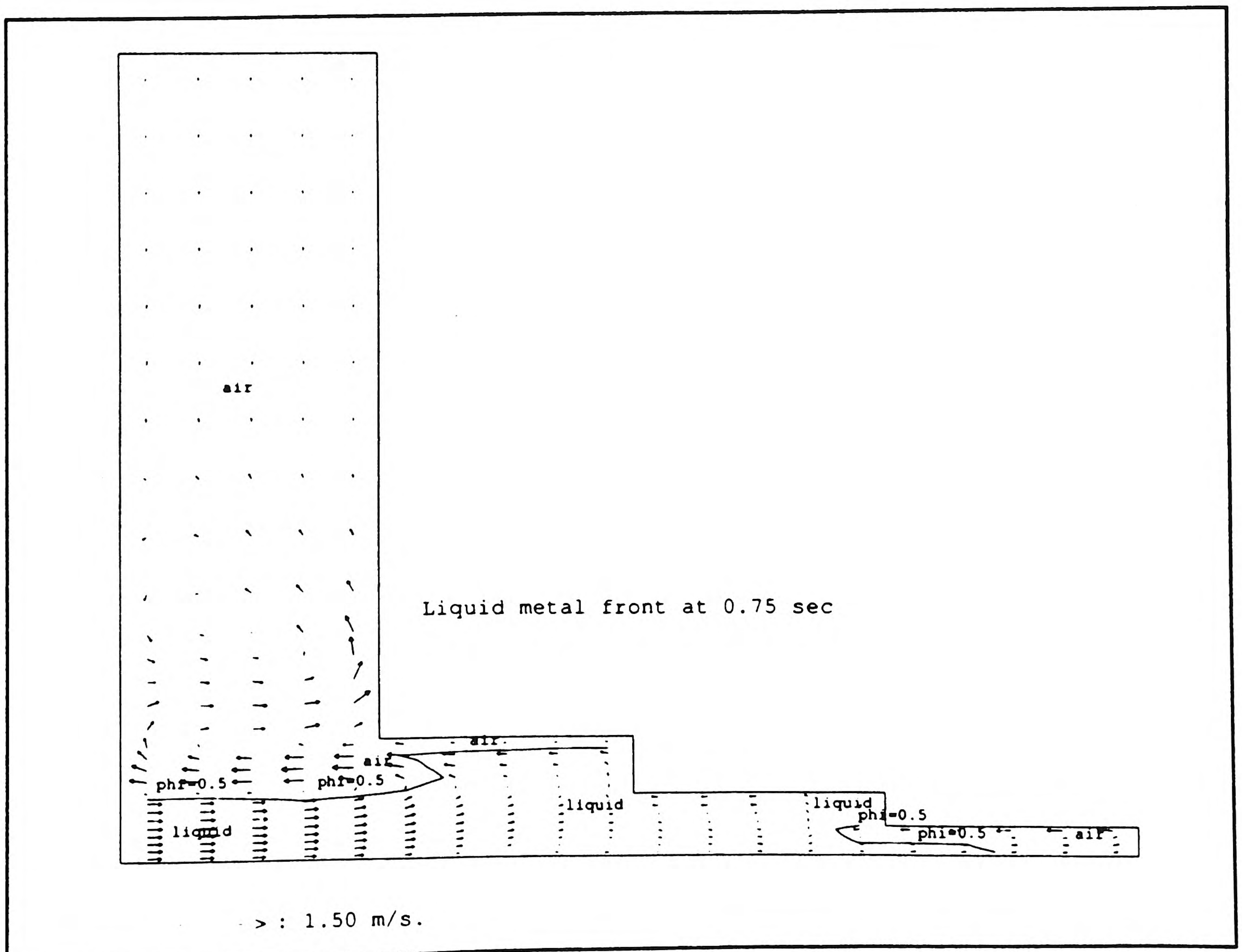


Figure 6.1b Velocity vectors and free surface after 0.75 second.

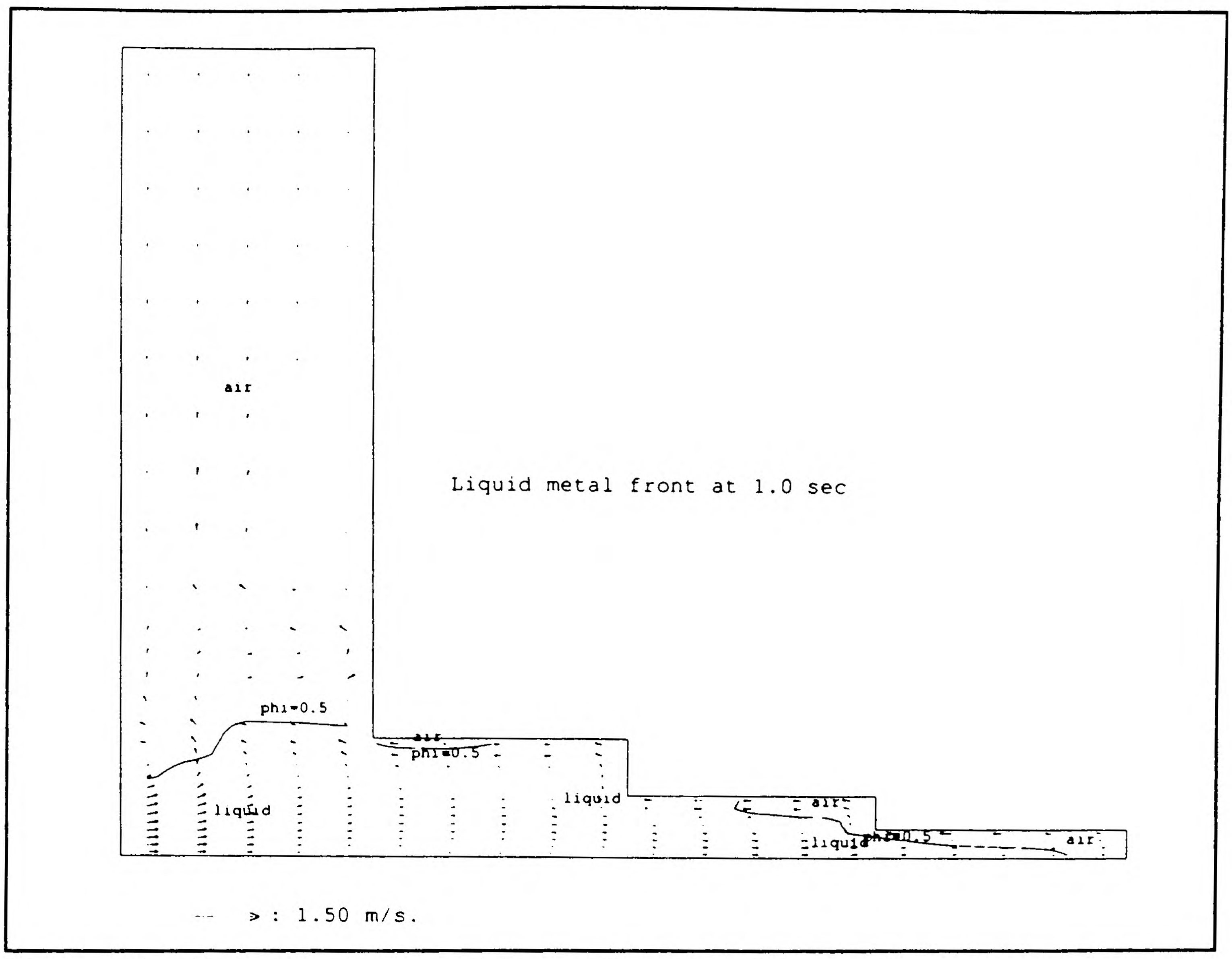


Figure 6.1c Velocity vectors and free surface after 1.0 second.

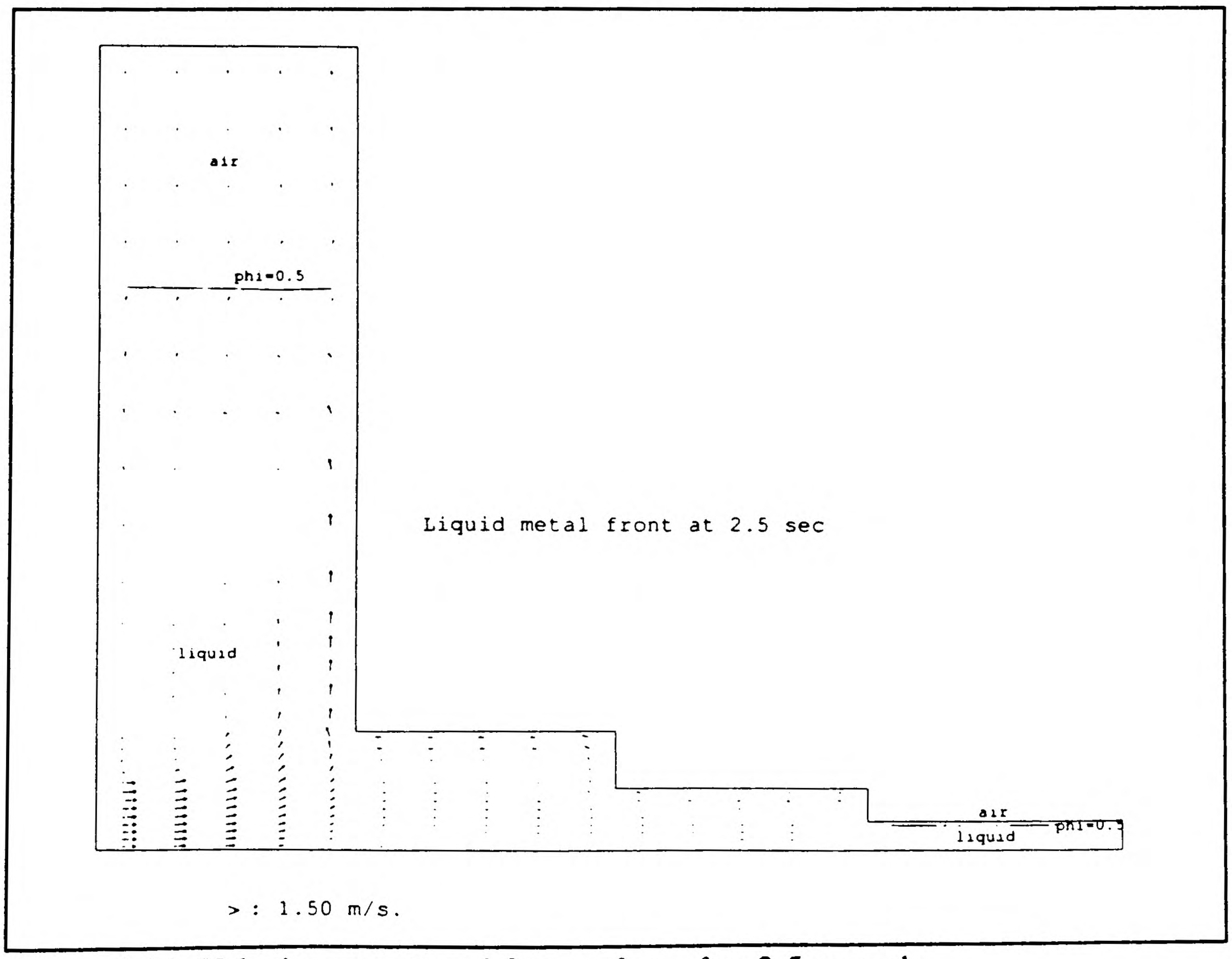


Figure 6.1d Velocity vectors and free surface after 2.5 seconds.

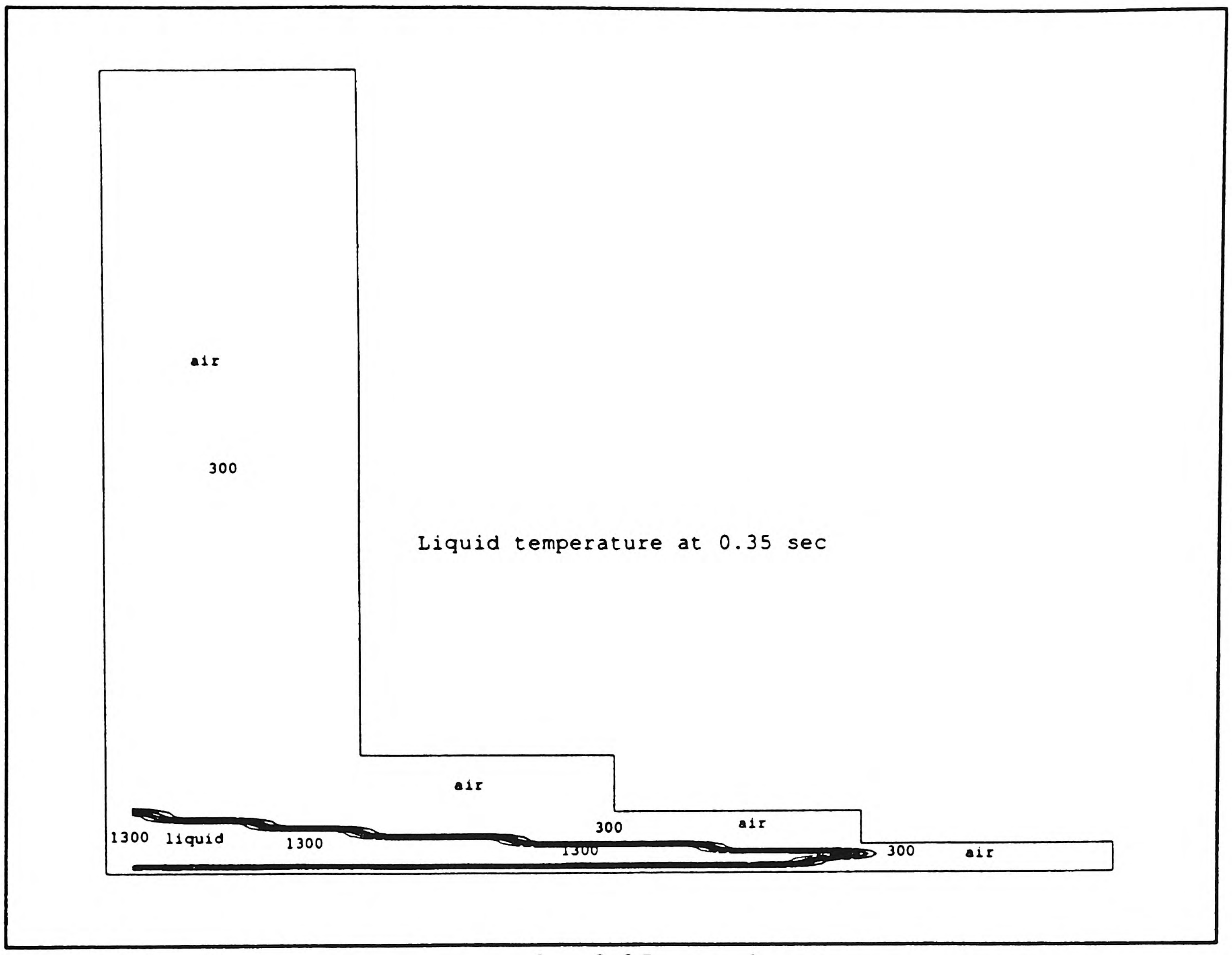


Figure 6.2a Temperature contours after 0.35 second.

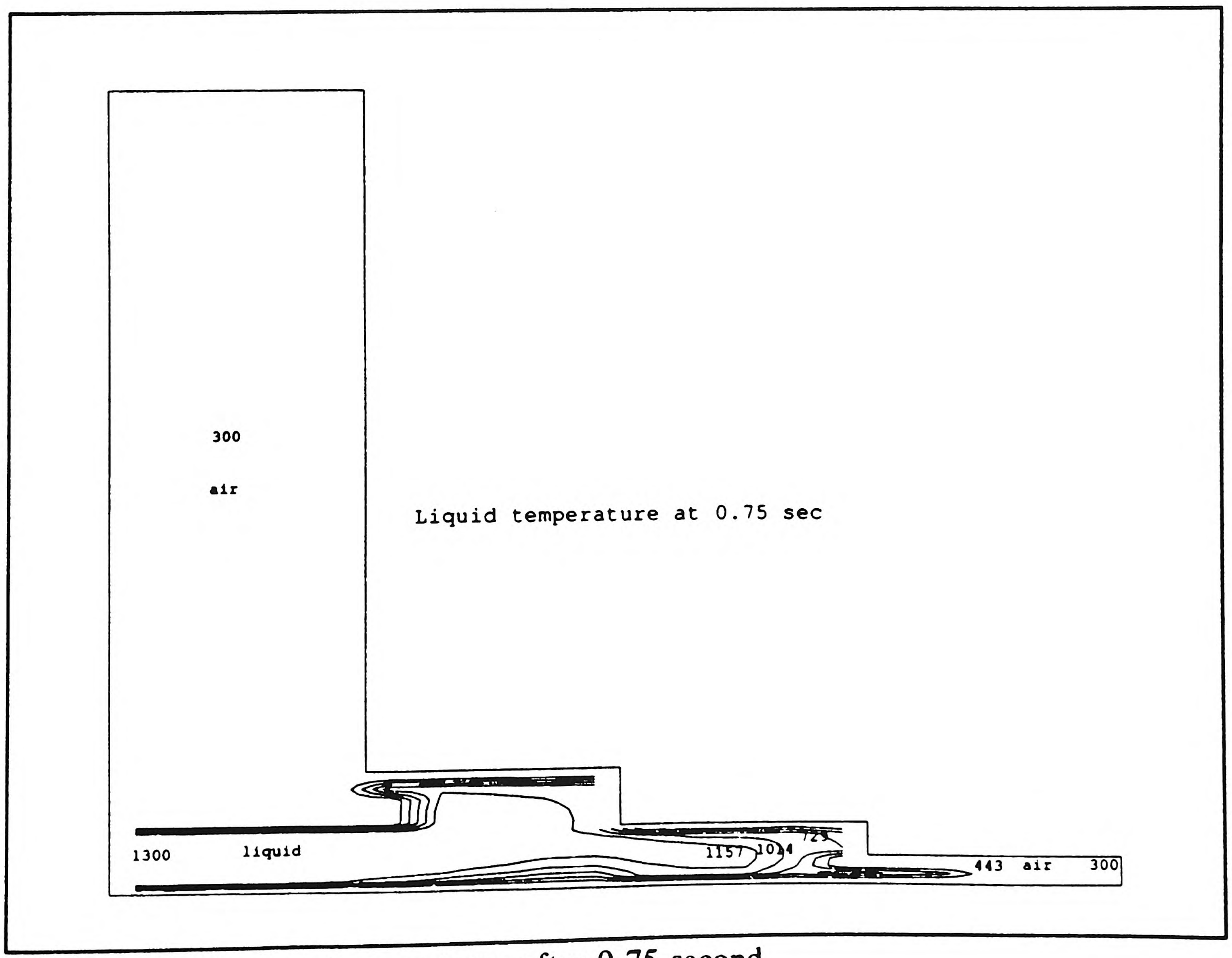


Figure 6.2b Temperature contours after 0.75 second.

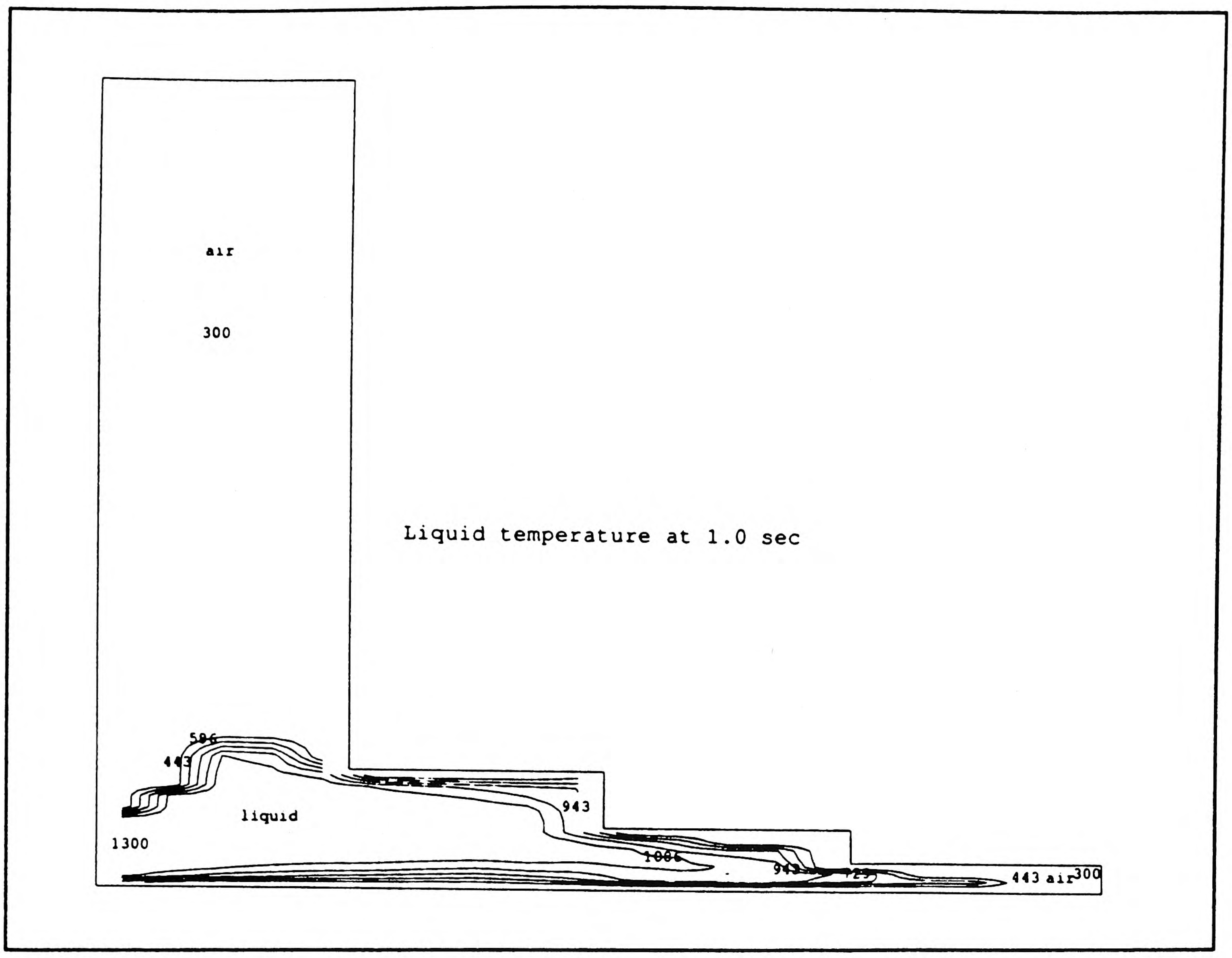


Figure 6.2c Temperature contours after 1.0 second.

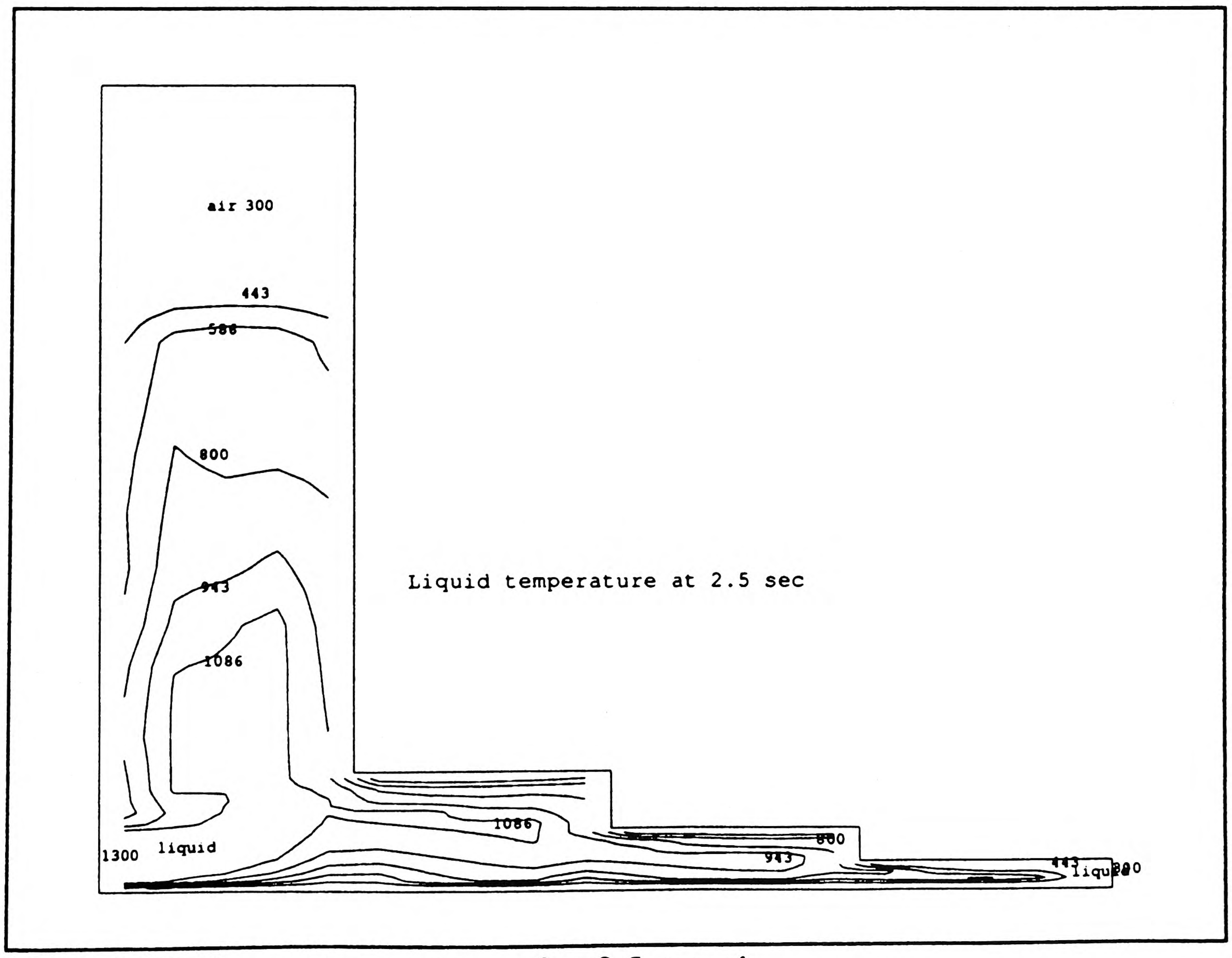


Figure 6.2d Temperature contours after 2.5 seconds.

takes place in the narrowest part of the liquid layer, due to convective recirculation. At 1.0 second, (*Figure 6.2c*), the liquid comes into contact with the top surface where more heat is lost through the mould. The temperature contours are now more orderly, with temperatures gradually diminishing from the bulk of the liquid towards its edges. This is a desirable temperature distribution, since closed contours (ie. pockets of hot liquid surrounded by a colder areas) may lead to void formation. By the time the feeder is nearly full, the main body of the casting (in the stepped region) has cooled down significantly. The hot liquid jet is now diverted upwards as indicated by the high temperature contours in *Figure 6.2d.*

6.4.3 Comments for Example I

The application of heat transfer with the SEA algorithm is demonstrated. The method is applicable to any free surface scenarios, involving two or more immiscible fluids. The results obtained for the case presented appear to be realistic and reasonable. However, the assumption of constant air temperature within the mould might not be appropriate if the effects of heat convection and radiation from the hot metal or from the pre-heated sand mould are taken into account. In that case the model might then be over complicated to an extent that proper two phase techniques such as IPSA [Spalding (1977, 1980a)] will have to be used. Experimental data is also required to validate this model.

6.5 Example Case II: Filling with Heat Transfer and Solidification - Step Wedge

The example to be presented in this section is similar in geometry to the step mould described in *Section 6.4*, though the dimension of the geometry is not the same, see *Figure 6.3* for details. A mesh of 20 x 20 = 400 cells was used for this simulation, see *Figure 6.4.*

6.5.1 Problem Specification

The inflow velocity and other initial and boundary conditions are identical to those described in *Section 6.4.1.* The initial liquid temperature is set to 1053°K. The solidus and liquidus temperatures for phase change are set to 913°K and 953°K respectively.

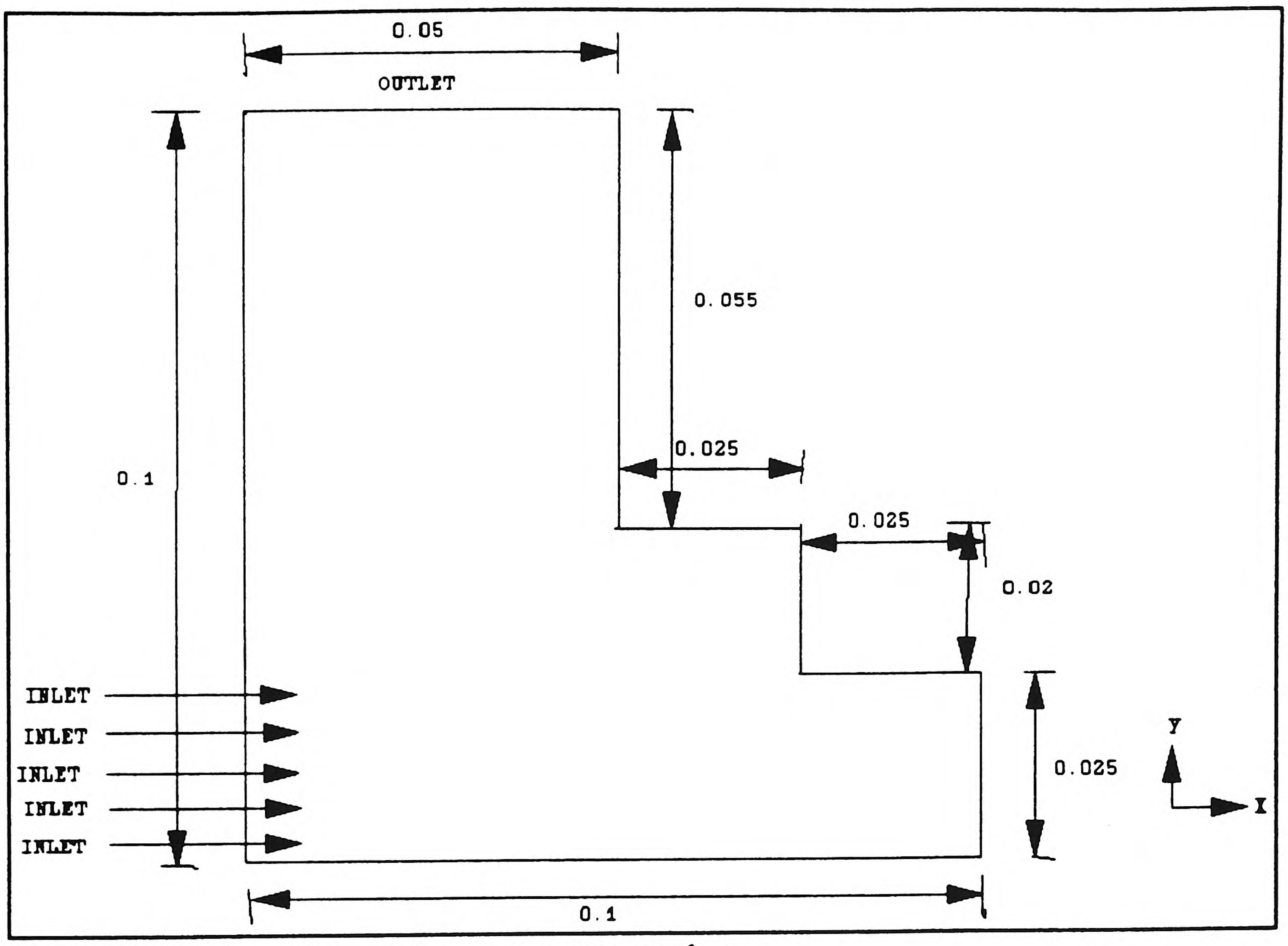


Figure 6.3 Dimension of the square step wedge.

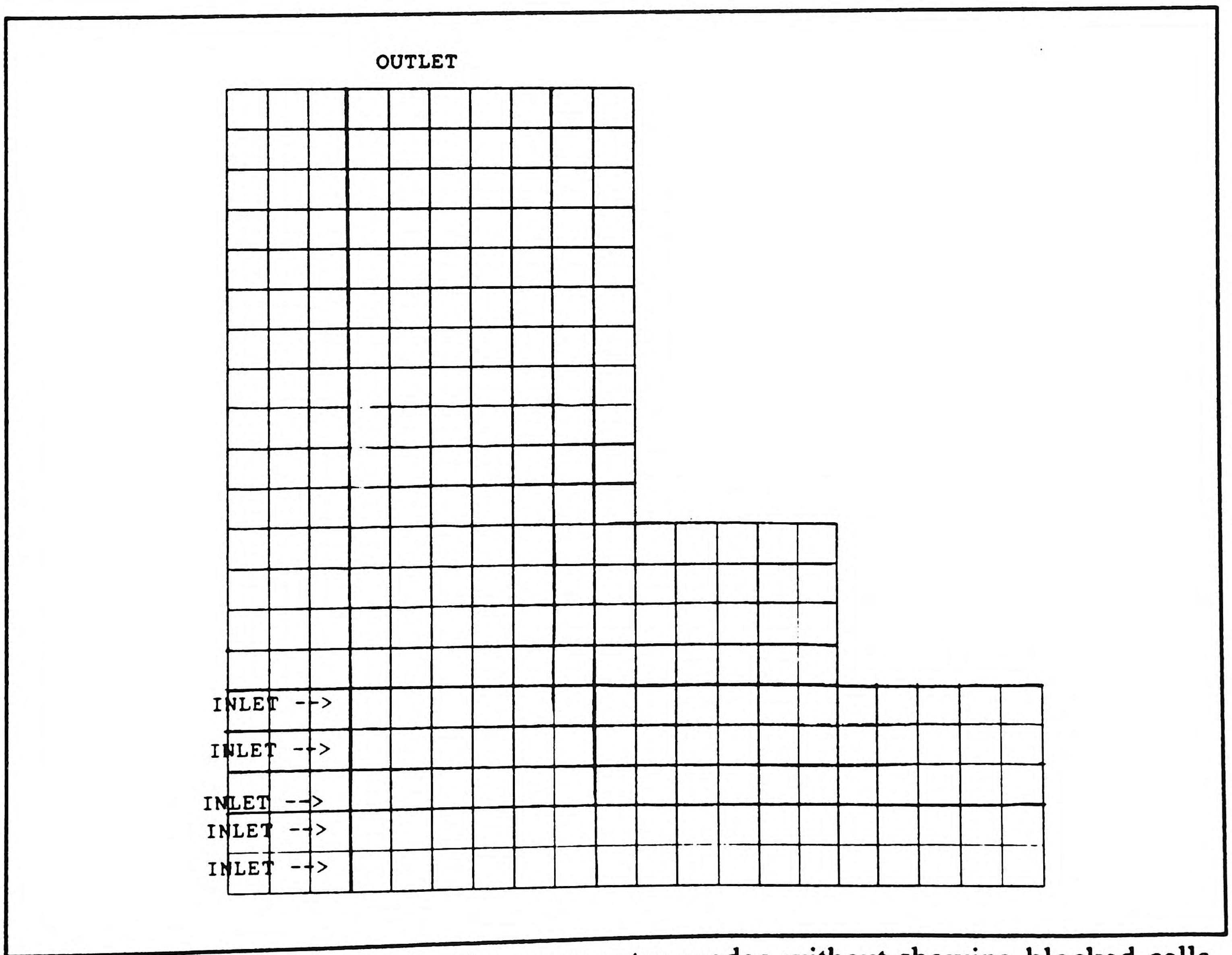


Figure 6.4 The mesh used for the square step wedge without showing blocked cells.

Also, the latent heat of phase change L is set to 2.8x10^5 J/kg.

The fluid properties for this simulation is listed in *Table 6.2* below:

Materials	Aluminium Alloy	Air	Sand
Density ρ, (kg/m^3)	2670.0	1.0	2000
Kinematic Viscosity ν,(m$^{2/s}$)	3.7x10^{-7}	2x10^{-5}	-
Specific Heat C_p, (J/kg°K)	880.0	993	676
Thermal Conductivity k, (W/m°K)	180.0	0.024	0.733

Table 6.2 Material properties for step mould.

6.5.2 Results and Remarks

The results of the simulation is given in *Figures 6.5(a to d).* These figures show that the filling process from 0.1 second to 0.5 second. At 0.5 second the feeder is nearly full of fluid. Plots are given in vector field, the free surface air-liquid interface (at ϕ= 0.5), and the boundary for solid/mushy regions (at f_l = 0.5) for four time intervals (0.1 second, 0.2 second, 0.3 second and 0.5 second).

Figure 6.5a (at 0.1 second), shows the liquid entering the mould cavity and reaching the far end of the first step. A wave is forming as the result of liquid impacting on the end wall, and air escaping from the cavity via the top of the feeder. As the hot molten liquid metal makes contact with the cold sand wall at the moment it enters the mould, a thin layer of solid/mushy zone is formed along the floor of the sand mould.

Figure 6.5b (at 0.2 second), shows that the first step is completely filled and a rather strong wave is moving away from the end wall of the second step and striking the opposite wall. At this point the solid/mushy region expands around the floor and far wall of the mould.

After 0.3 second, see *Figure 6.5c*, more liquid reaches the mould, and the already

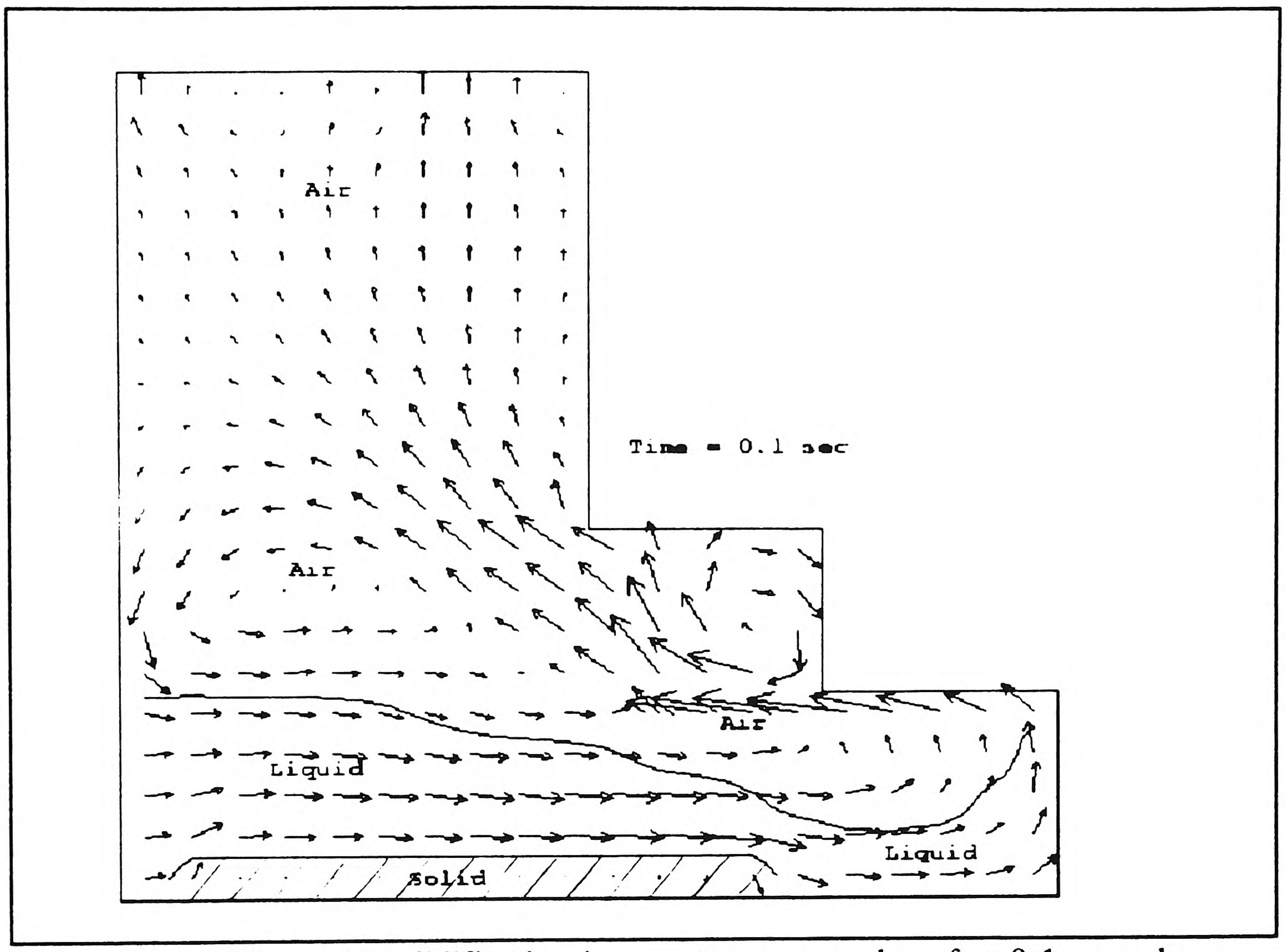


Figure 6.5a Filling and solidification in a square step wedge after 0.1 second.

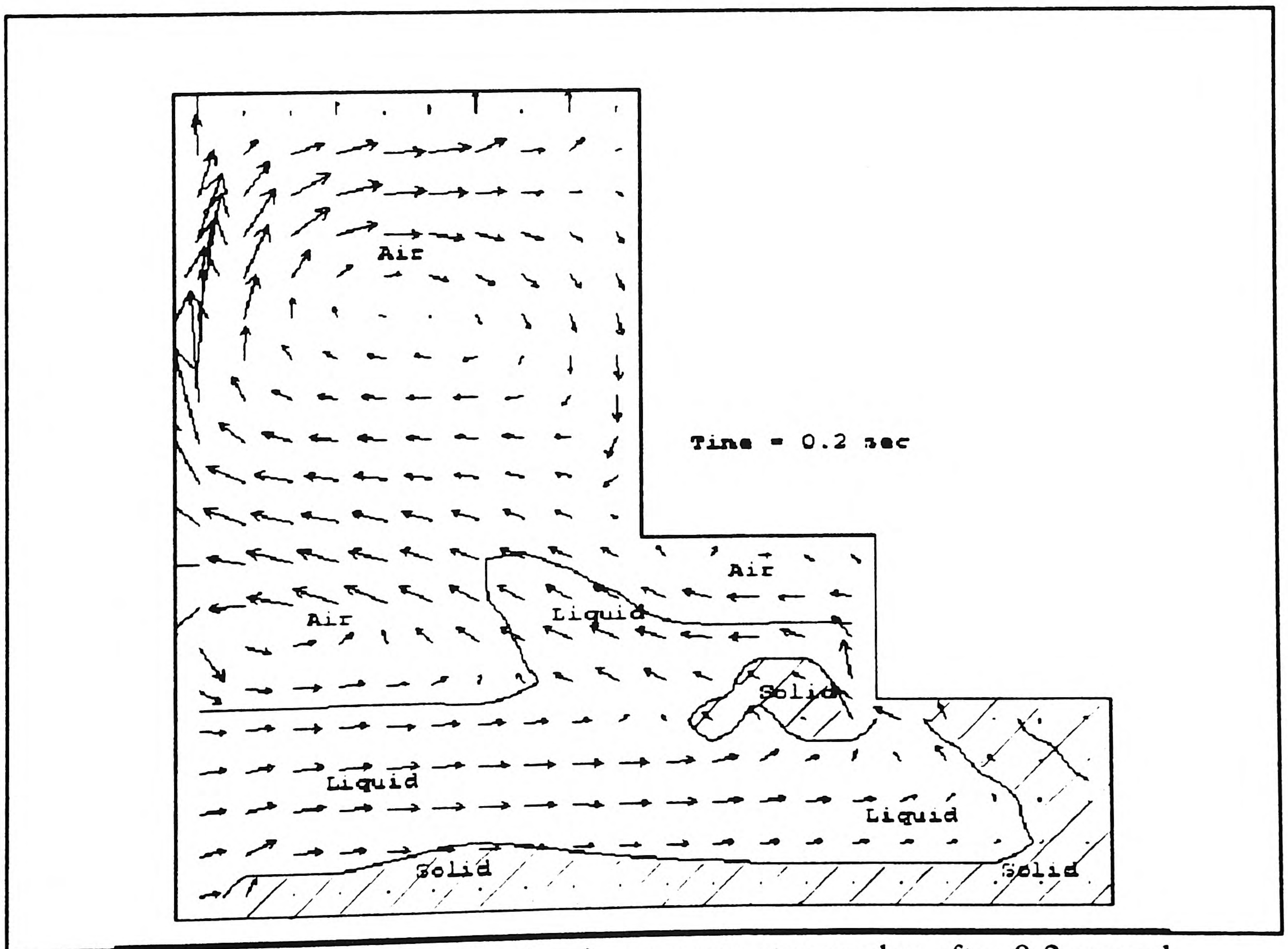


Figure 6.5b Filling and solidification in a square step wedge after 0.2 second.

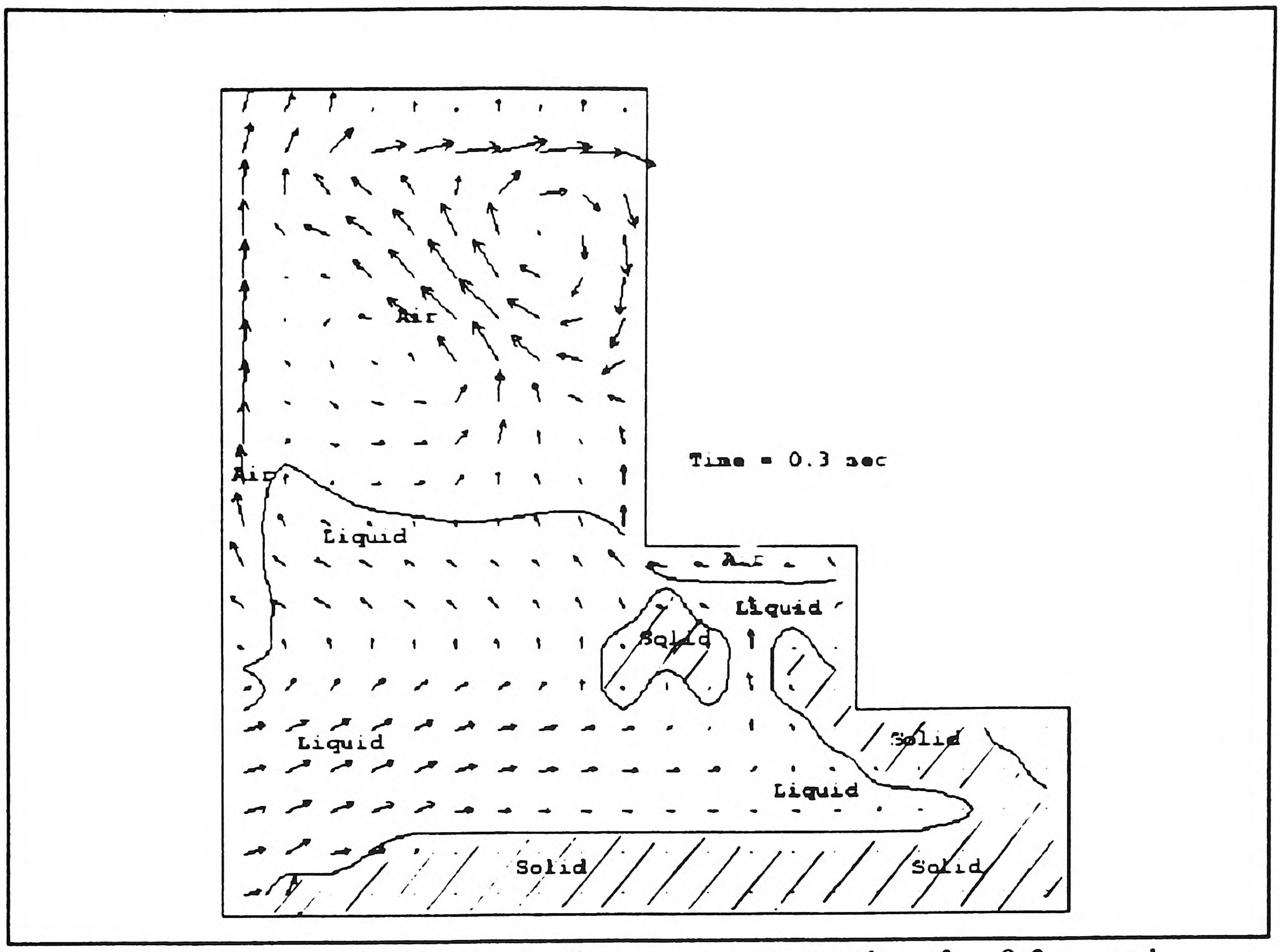


Figure 6.5c Filling and solidification in a square step wedge after 0.3 second.

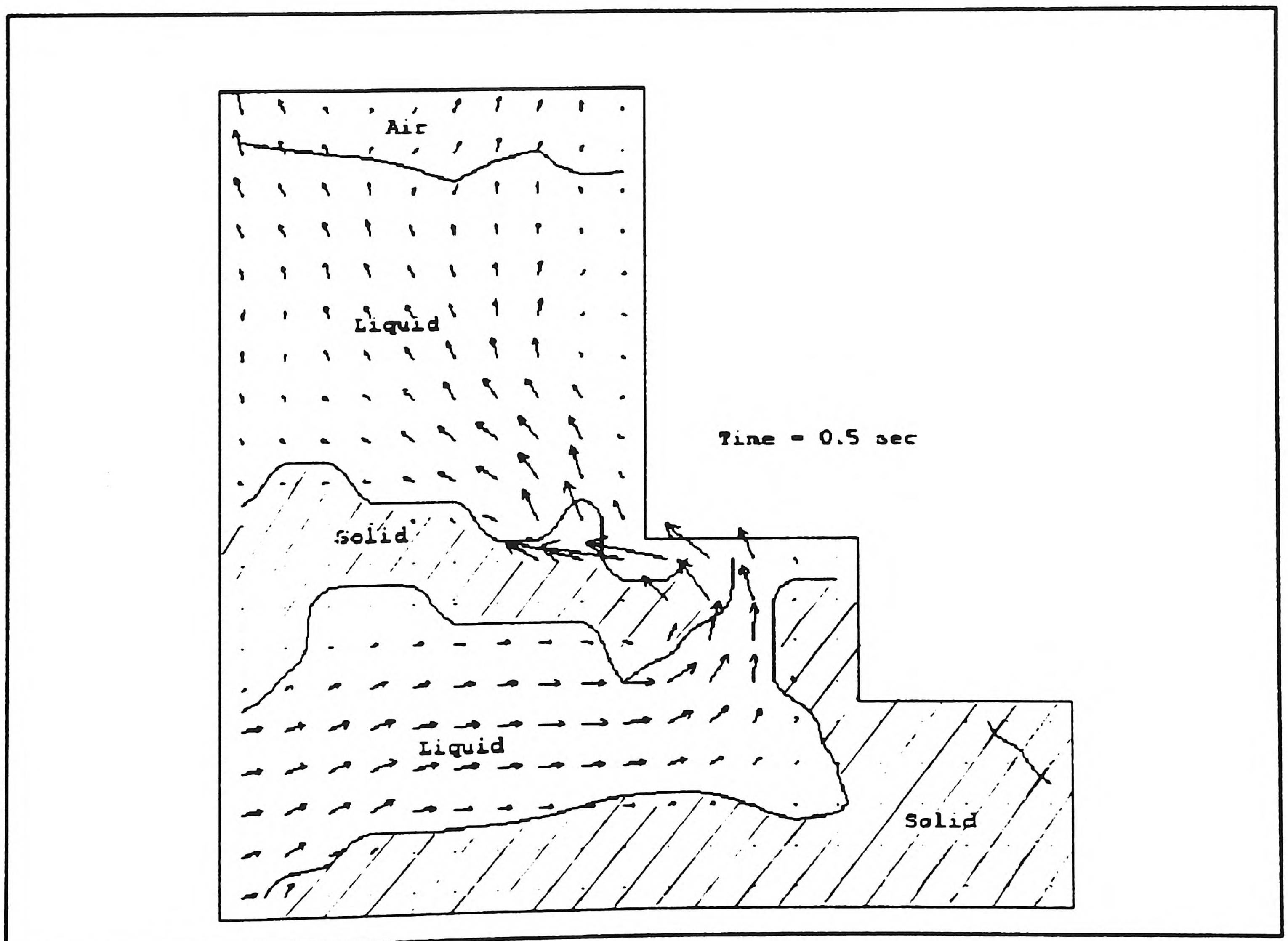


Figure 6.5d Filling and solidification in a square step wedge after 0.5 seconds.

solidified regions are getting thicker due to the cooling effect. However, some re-melting occurs within the second step. Also, note that a small solid/mushy region has formed just above the ingate area. At 0.5 second (*Figure 6.5d*), the mould is almost completely filled and most of the first step is completely solidified. In addition, a solid/mushy region extending from the left hand wall of the feeder is approaching the mushy region which formed at the end wall of the second step. As a consequence, large velocity vectors are found in this area as the liquid trying to flow through the closing 'gap'.

6.5.3 Comments for Example II

The introduction of solidification has demonstrated the significant effect that the mushy zone has on the dynamics of free surface motion, and on the temperature distribution in the liquid. It is concluded that in a process where rapid solidification occurs, such as casting molten alloys into permanent metal moulds, as for example, Gravity and Pressure-Die Casting, the filling and solidification processes have to be modelled simultaneously. Work on improvement in temperature estimation and validation is vital for this model to be successful.

6.6 Closure

The formulation of both heat transfer and solidification and implementation into the SEA algorithm has been illustrated in this Chapter. The impact of such model must not be undervalued, since many subsequent physical phenomena which occur in casting processes are directly affected by the fluid dynamics during mould filling [Campbell (1991)]. These physical phenomena include oxide-film formation, shrinkage, macro- and micro-segregation, residual stress and deformation (which leads to the formation of air-gap), and macro- and micro-porosities formation.

Chapter 7

CONCLUDING REMARKS

7.1 Conclusions

In this study, the development and examination of the Scalar Equation Algorithm (SEA) to model mould filling with phase-change in three dimensions has been addressed.

By and large, the SEA filling algorithm has shown promising potential in modelling general filling processes, whether it is under high pressure or under natural gravity. Furthermore, it has great significance towards economy of computing time and resources. Without such a technique being developed, one may have to resort to two-phase calculations, such as IPSA [Spalding (1977, 1980a)] to obtain the appropriate liquid temperature. As a consequence, the number of equations to be solved will be doubled.

The ensuing incorporation of solidification functionality into the model enables the simulation of filling and phase change processes to be done simultaneously. It forms the link between free surface filling and subsequent physical phenomena, such as stress analysis and porosity formation, which occur in the casting process. This points to the good prospect of creating an integrated casting simulation software for the foundry industry.

The major findings from this study are summarised as follows:

- A three dimensional free surface model based on a scalar advection equation [Liu Jun (1985, 1986b)] expressed in terms of volumetric continuity with the aid of GALA [Spalding (1974, 1977)], using the van Leer scheme [van Leer (1977)] to suppress numerical diffusion, has been formulated and implemented.

- The model, without heat transfer and phase change, has been validated against experimental data from both actual sand mould casting and water experiments. In addition, the capability of the model in capturing complex flow patterns, which include convoluted wave formation and the entrainment of air and impurities have been demonstrated conclusively using geometries supplied by industry.

• The SEA filling algorithm has been successfully implemented into another structured control-volume based CFD code - Harwell-FLOW3D. This has shown that the algorithm itself is formulated in a generic fashion rather than having a bias towards PHOENICS, making it portable in nature. Therefore, it is possible to implement it into unstructured control-volume based CFD codes, such as ASTEC [AEA Technology] and UIFS [Chow (1993)]. In fact, the filling algorithm already implemented in ASTEC is based on a similar idea to the SEA algorithm, except that it uses a heuristic rule-based scheme to keep the free surface sharp.

• To alleviate the restriction of the Courant criterion on the time step used in the simulation, the Harwell-FLOW3D version of SEA filling model is capable of running in parallel either on a series of INMOS transputers or a set of i860-transputer vector processor modules. A computational efficiency of up to 80% over the serial version has been achieved, depends on the number of processors and the size of the problem domain (in terms of computational cells or elements).

• A technique to model heat transfer between two immiscible fluids within a single phase framework has been developed. Although no validation has been undertaken due to lack of published experimental data, the preliminary results obtained by the technique are reasonable and promising.

• Incorporation of the solidification model [Voller and Prakash (1987b)] into the SEA filling algorithm with heat transfer, enhanced the model's capability to simulate not just air and liquid but also, solid and mushy zones, formed due to phase change. All these different 'phases' are simulated by a set of momentum and energy equations in a single phase framework.

• Again due to lack of experimental data, the filling with phase change simulation has not been validated. However, the preliminary results were encouraging.

7.2 Suggestions for Future Work

The author would like to give suggestions on three different aspects as follows:

(I) From the aspect of computational efficiency

The major cause that prevents virtually all current filling simulation software to become day-to-day tools used by the foundry industry in the UK and abroad is that they are very expensive in two factors:

1. Cost of the software and hardware.
2. Computing time required.

The first factor is understandable, as the capital costs to develop new software are high, especially for engineering software. However, if the number of clients increases to a level that the costs can be shared out more evenly, and with more competition from CFD software houses, the price of such software will fall just like those in the commercial sectors. The hardware required to run simulations is dictated, to an extent, by the developer of the software concerned. Nevertheless, it is the second factor that deters most medium to small foundries to buy such software; of course the factor of knowledge and expertise to operate such software must also be taken into consideration. No industrial users would prepare to wait days for a simulation results, hours (within a day) may be what is tolerable by them. They would rather utilize the time to perform more trial 'runs' of the casting of interest with different runner and feeder arrangements. Alternatively, they may also use empirically based simulation software such as SOLSTAR and FeederCalc to model the optimum gating arrangement for the casting, in order to achieve the required casting quality.

The author has investigated using variable time stepping to overcome the Courant restriction in this study but without significant run-time benefits. An alternative is to reformulate the SEA algorithm if possible, so that it is implicit in time. Partial solutions to this problem have been suggested by Voller and Swaminathan (1993) and Liu Jun and Spalding (1992) respectively. Unfortunately, at the time of writing this report, the enthalpy based VOF method (EVOF) still cannot handle emptying of a cell after it has been filled. In other words, the model cannot capture waves or sloshing of the fluid. The Height of Liquid (HOL) method goes a step further. It can handle filling and emptying of a cell but not sloshing.

(II) From the aspect of numerical accuracy and validation

The sharpness at the air/liquid interface must be retained all the times, but unfortunately some degree of numerical smearing still exists with the SEA algorithm. This is due to the very nature of the scalar advection equation. The use of a concentration factor to estimate the proportion of one fluid from a mixture always result in some error. To maintain a distinctive interface between the two fluids, the 'slip' algorithms such as those reported by Pericleous and Drake (1985) and Chellam et al (1992), may be adopted into the SEA algorithm to separate the heavier liquid particles from that of air in a mixed cell.

Alternatively, a new time-implicit method, similar to the way Voller [Voller et al (1987a)] handles solidification problem may be able to solve the smearing problem. This new method to eliminate smearing at the air-liquid interface (in slow moving fluids) would probably adopt the ideas suggested by either the EVOF or HOL method. The new technique must not have the same defects associated with its predecessors.

In order to win over the mind of sceptical practitioners from the foundry industry, the filling model with heat transfer and phase change must be subjected to stringent validation. However, the major difficulty in this area is that good industrial scale experimental data is scarce and patchy. Collaboration with the foundry industry and/or other academic institutions which specialise in foundry technology may be a solution. For example, the ACME casting initiative project currently running at the University of Greenwich, is jointly funded by the DTI, SERC and industrial sponsors, such as Rolls Royce, British Aerospace and AEA Technology.

(III) From the aspect of applications and capabilities

The current method is based on a control-volume structured grid which hinders the kind of geometry that can be simulated by the model. Furthermore, the use of blockages to create the necessary shape of the mould is expensive in terms of computing storage and CPU time. On the other hand, finite element based methods have no such problem. So one suggestion would be to implement the SEA algorithm within the framework of unstructured, non-staggered CV based CFD code (such as UIFS [Chow (1993)] and ASTEC [AEA Technology]) to ensure its future and to enhance its application capability. A direct prerequisite to provide such capability is that a built-in ability to generate unstructured grids in three-dimensions must at hand. Or at least, the functionality to be able to interface with other CAD and/or mesh generation packages (such as FAM, PATRAN, ANSYS, ARIES and AutoCAD) should be addressed.

With respect to the capability side of the model, the author suggests the following:

1) Incorporation of a radiation model to calculate heat losses due to thermal radiation is particularly important in the simulation of investment casting.
2) Incorporation of surface tension effects to simulate the filling of thin section castings and the filling of polymer other than metals.
3) Use of the Bernoulli equation to calculate inflow pressure, such that a variable inflow velocity depending on the external and internal pressure can be modelled.
4) Include permeability at sand walls to allow air to escape. (The author has attempted to introduce air permeability at sand moulds but the results were not encouraging).
5) Coupling with other physical modules which have been developed as a part of the casting initiative to form an integrated casting simulation software. These physical modules include residual stress and deformation [Bailey C et al (1993)] (which can model the formation of air-gaps during pure solidification after the mould is fully filled) and porosity formation modelling [Fryer et al (1993)].

REFERENCES

3L Limited
Parallel Fortran User Guide (version 2.1.2)
3L Ltd, Livingston, Scotland.

AEA Technology
(FLOW3D and ASTEC CFD codes)
AEA Technology, Harwell, Oxon, UK.

AFS Cast Metals Technology Series (1976a)
Basic Principles of Gating
American Foundrymen's Society Training and Research Institute, Cast Metals Technology Series, Publisher: Addison-Wesley, USA, 1976.

AFS Cast Metals Technology Series (1976b)
Basic Principles of Rising
American Foundrymen's Society Training and Research Institute, Cast Metals Technology Series, Publisher: Addison-Wesley, USA, 1976.

Aldham C, Rhodes N and Tatchell D (1982)
Three-dimensional Calculations of Explosion Containment in Fast Breeder Reactors
Pub. 82-FE-3, ASME, Fluids Eng Division, 1982.

Alphacast Software
(MAVIS and DIANA - rule based lattice concept casting simulation software products)
Alphacast Software Limited, Northampton, UK.

Aluminium Pechiney
(SIMULOR - a CFD based casting simulation software)
Aluminium Pechiney, Voreppe, France.

Amsden A A and Harlow F H (1970)
The SMAC Method: A Numerical Technique for Calculating Incompressible Fluid Flows
Report LA-4370, Los Alamos Scientific Laboratory, USA, 1970.

Anzai K and Uchida T (1991)
Mold Filling Patterns of Flat Plat Die Castings
Modeling of Casting, Welding and Advanced Solidification Processes V
Editors: Rappaz M, Ozgu M R and Mahin K W, Publisher: The Minerals, Metals & Material Society, 1991, pp. 741-748.

Amato M, Matrone A and Schiano P (1994)
A Practical Experience in Parallelizing a Large CFD Code: The ENSOLV Flow Solver
High-Performance Computing and Networking
Editors: Gentzsch W and Harms U, Publisher: Springer-Verlag
Proceedings Volume II: Networking and Tools, 1994.

Bailey C, Fryer Y D, Cross M and Chow P (1993)
Predicting the Deformation of Castings in Moulds using a Control Volume Approach on Unstructured Meshes
Mathematical Modelling for Materials Processing
Editors: Cross M, Pittman J F T and Wood R D, Publisher: Oxford University Press, USA, 1993, pp.259-272.

Backer G P and Sant F J (1991)
Fluid Flow Analysis of Pinholes in a Malleable Iron Casting
Numerical Simulation of Casting Solidification in Automotive Application
Editors: Kim C and Kim C-W, Publisher: The Minerals, Metals and Material Society, 1991, pp. 189-205.

Booth S E and Allsop D F (1983)
Cavity Flow Studies at BNF Metals Technology Centre Using Water Modelling Techniques
Paper presented to the 12th International Die Casting Congress & Exposition, Minneapolis, MN, USA, 1983,
Paper No: G-T83-033, The Society of Die Casting Engineers, IL, USA, 1983.

Bull J M (1990)
Recent Development in Numerical Methods for the Advection Equation and their Application to Boundary Layer Modelling
Confidential Report; Turbulence and Diffusion Note 195, Meteorological Office, Bracknell, UK, 1990.

Burns A D and Wilkes N S (1987)
A Finite Difference Method for the Computation of Fluid Flows in Complex Three Dimensional Geometries
AERE - Report. 12342, AEA Technology, Harwell Laboratory, Oxon, UK, (1987)

Campbell J (1991)
Castings
Butterworth-Heinemann Ltd, UK, 1991.

Castrejon A (1984)
Unsteady Free-convection Heat Transfer inside an Annular Cavity
PhD Thesis, University of London, 1984.

Casulli V and Greenspan D (1982)
Numerical Solution of Free Surface, Porous Flow Problems
Int. J. Numerical Methods in Fluids, 1982, vol. 2, pp. 115-122.

Codina R, Schafer U and Onate E (1992)
Mould Filling Simulation Using Finite Elements
Publication No. 24, International Centre for Numerical Methods in Engineering, Gran Capitan s/n, 08034 Barcelona, Spain.

CHAM Limited
(PHOENICS CFD general simulation code)
CHAM Limited, Wimbledon, London, UK.

Chan K S, Pericleous K and Cross M (1991)
Numerical Simulation of Flows Encountered During Mold-Filling
Appl. Math. Modelling, 1991, vol. 15, Nov/Dec, pp.624-631.

Chan R K-C and Street R L (1970)
Computer Studies of Finite-Amplitude water Waves
J. Comp. Phys., 1970, vol. 6, pp. 68.

Chellam S, Wiesner M R and Dawson C (1992)
Slip at a uniformly porous boundary: effect on fluid flow and mass transfer
Journal of Engineering Mathematics, 1992, vol. 26, pp 481-492.

Chen L and Vorus W S (1992)
Application of a Vortex Method to Free Surface Flows
Int. J. for Numerical Meth. in Fluids, 1992, vol. 14, pp. 1289-1310.

Chorin A J (1980)
Flame Advection and Propagation Algorithms
J. Comp. Phys., 1980, vol. 35, pp. 1-11.

Chow P and Cross M (1992)
An Enthalpy Control-Volume - Unstructured Mesh (CV-UM) Algorithm for Solidification by Conduction only
Int. Journal for Numerical Methods in Engineering, 1992, vol. 35, pp. 1849-1870.

Chow P M Y (1993)
Control Volume Unstructured Mesh Procedure for Convection-Diffusion Solidification Process
PhD Thesis, Sch. of Maths. Stats & Comp., University of Greenwich, London, 1993.

Courant R, Isaacson E and Rees M (1952)
On the solution of nonlinear hyperbolic differential equations by finite differences
Comm. Pure Appl. Math., 1952, vol. 5, pp. 243-255.

Crank J (1984)
Free and Moving Boundary Problems
Oxford University Press, NY, USA, 1984.

Cross M, Johnson S P, Leggett P F and Ierotheou C S (1994)
The Computer Aided Parallelisation Tools 1: Overview of an Interactive Toolkit For Mapping FORTRAN Code Onto Parallel Architectures
1994, (In Press).

Daniels H, Peters A, Bergen O, and Forkel C (1994)
Large Implicit Finite Element Simulations of Time-Dependent Incompressible Flows on Scalable Parallel Systems
High-Performance Computing and Networking
Editors: Gentzsch W and Harms U, Publisher: Springer-Verlag
Proceedings Volume II: Networking and Tools, 1994, pp. 119.

de Vahl Davis G (1983)
Natural convection of air in a square cavity: a benchmark numerical solution
Int. J. Num. Methods Fluids, 1983, vol. 3, pp. 249-264.

Denton B (1990)
Private communication, BNF-Fulmer, Oxford, 1990.

Denton B (1991)
Private communication, BNF-Fulmer, Oxford, 1991.

Deville M O (1974)
Numerical Experiments on the MAC Code for a Slow Flow
J. Comp. Phys., 1974, vol. 15, pp. 362-374.

Dewtec Systems
(AF SOLID and SYNERGY SOLIDIFICATION SYSTEM 3-D - CFD casting simulation software products)
Dewtec System Limited, Birmingham, UK.

Dhatt G, Gao D M ,and Ben Cheikh A (1990)
A Finite Element Simulation of Metal Flow in Moulds
Int. J. for Numerical Meth. in Eng., 1990, vol. 30, pp. 821-831.

Dufort E C and Frankel S P (1953)
Stability conditions in the numerical treatment of parabolic differential equations
Mathematical Tables and Other Aids to Computations, 1953, vol. 7, pp. 135.

Harwell-FLOW3D User Manual (1991)
FLOW3D User Manual (version 2.3.3)
AEA Technology, Harwell Laboratory, Oxon, UK, 1991.

Flow Science Inc
(FLOW-3D - a commercial filling simulation software code based on the VOF method)
Flow Science Inc, 1325 Trinity Drive, Los Alamos, New Mexico, 87544, USA.

FOSECO Limited
(FeederCalc - an empirically based feeder and runner design software tool)
FOSECO Limited, UK.

FOSECO and Micromet
(SOLSTAR - empirically rule based lattice concept casting simulation software)
Developed by Micromet and marketed by FOSECO Limited, UK.

Foundry Computational Service
(MATRIX - a CFD casting simulation software product)
Innovation Centre, University College of Swansea, Swansea, UK.

Frommer L 1933
Handbuch der Spritzgusstechnik
Julius Springer, 1933, s.20

Fryer Y D, Bailey C, Cross M and Chow P (1993)
Predicting Macro Porosity in Shape Casting Using an Integrated Control Volume Unstructured Mesh (CV-UM)
<u>Modeling of Casting, Welding and Advanced Solidification Processes VI</u>
Editors: Piwonka T S, Voller V and Katgerman L, Publisher: The Minerals, Metals & Materials Society, Pennsylvania, USA, 1993, pp. 143-151.

Galea E, Chan A (K S), Cross M, Hoffmann N, Ierotheou C, Johnson S and Pericleous K (1993)
Application of a Parallel CFD code to Large-Scale Practical Problems
<u>Parallel Computational Fluid Dynamics '92</u>
Editors: Pelz R B, Ecer A and Hauser J, Publisher: Elsevier Science Publisher B V, (North Holland), 1993.

Golafshani M (1988)
A Simple Numerical Technique for Transient Creep Flows with Free Surfaces
Int. J. for Numerical Meth. in Fluids, 1988, vol. 8, pp. 897-912.

Goodway M (1988)
History of Casting
<u>Metals Handbook Ninth Edition, Volume 15 Casting</u>
Publisher: ASM INTERNATIONAL Handbook Committee, Ohio, USA, 1988, pp. 15-23.

Haberhauer S (1994)
Unstructured Computational Fluid Dynamics on Massively-Parallel SIMD-Systems
<u>High-Performance Computing and Networking</u>
Editors: Gentzsch W and Harms U, Publisher: Springer-Verlag
Proceedings Volume II: Networking and Tools, 1994, pp. 131.

Harlow F H and Welch J E (1965)
Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surfaces, Phys. Fluids, 1965, vol. 8, pp. 2182.

Harlow F H and Nakayama P I (1967)
Turbulent Transport Equations
Phys. Fluids, 1967, vol. 10, pp. 2323-2332.

Harwell-FLOW3D
AEA Technology, Harwell, Oxford, UK.

Hayes F (1989)
Intel's Cray-on-a-Chip
BYTE, 1989, vol. 14, May, pp 113-114.

Hirt C W and Shannon J P (1968)
Free-Surface Stress Conditions for Incompressible flow Calculations
J. Comp. Phys., 1968, vol. 2, pp. 403-411.

Hirt C W and Cook J L (1972)
Calculating Three-Dimensional Flows around Structures and over Rough Terrain
J. Comp. Phys., 1972, vol. 10, pp. 324-340.

Hirt C W, Nichols B D and Romero N C (1975)
SOLA - A Numerical Solution Algorithm for Transient Fluid Flows
Los Alamos Scientific Laboratory Report LA-5852, 1975.

Hirt C W and Nichols B D (1981)
Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries
Journal of Computational Physics, 1981, vol. 39, pp. 201-225.

Hoffmann N A (1990)
Computer Simulation of Fire-Sprinkler Interaction
PhD Thesis, University of Greenwich, London, UK, 1990.

Hwang J D, Lin H J and Su K C (1991)
Mathematical Modeling of Fluid Flow During Vertical Centrifugal Casting
Modeling of Casting, Welding and Advanced Solidification Processes V
Editors: Rappaz M, Ozgu M R and Mahin K W, Publisher: The Minerals, Metals and Materials Society, 1991, pp. 755-761.

Hwang W S and Stoehr R A (1983)
Fluid Flow Modeling for Computer Aided Design of Castings
Journal of Metals, 1983, vol. 35, pp. 22-29.

Hwang W S and Stoehr R A (1987)
Computer Simulation for the Filling of Castings
American Foundrymen's Society Transactions, 1987, vol. 141, pp. 425-430.

Hyman J M and Larrouturou B (1982)
The Numerical Differentiation of Discrete Functions Using Polynomial Interpolation Methods
Appl. Math. and Comp., 1982, vol. 10-11, pp. 487-506.

Ierotheou C S and Galea E R (1992)
A Fire Field Model Implemented in a Parallel Computing Environment
Intl. J. for Numerical Methods in Fluids, 1992, vol. 14, pp. 175-187.

Issa R I (1985)
Solution of the Implicitly Discretised Fluid flow Equations by Operator Splitting
J. Comp. Phys., 1985, vol. 61, pp. 40.

Jones I P, Kightley J R, Thompson C P and Wilkes N S (1985)
FLOW3D, A computer code for the prediction of laminar and turbulent heat transfer: Release 1
Atomic Energy Research Establishment Report 11825, UK, 1985.

Jones I P, Kightley J R, Thompson C P and Wilkes N S (1986)
FLOW3D release 1: User Manual
AERE R 11893, UK, 1985.

Johnson S P (1992)
Mapping Numerical Software Onto Distributed Memory Parallel Systems
PhD Thesis, University of Greenwich, London, 1992.

Johnson S P (1993)
Private communication, University of Greenwich, London, 1993.

Johnson S P and Cross M (1991)
Mapping Structured Grid 3D CFD codes onto Parallel Architectures
Appl Math Modelling, 1992, vol. 15, pp. 394-405.

Johnson W E (1970)
Development and Application of Computer Programs Related to Hypervelocity Impact
Systems Science, and Software report 3SR-353, 1970.

Jones B W, Everett M G and Cross M (1993)
Mapping Unstructured Mesh CFD Codes Onto Local Memory Parallel Architectures
Parallel Computational Fluid Dynamics '92
Editors: Pelz R B, Ecer A and Hauser J, Publisher: Elsevier Science Publisher B. V., 1993, pp. 231-240.

Kobe Steel
(CASTEM - a CFD casting simulation software product)
Kobe Steel Limited, Nishi-Ku, Kobe, Japan.

Kondic V (1968)
Metallurgical Principles of Founding
Publisher: Edward Arnold Ltd, UK, 1968.

Lemos C M (1992)
A Simple Numerical Technique for Turbulent Flows with Free Surfaces
Int. J. for Numerical Meth. in Fluids, 1992, vol. 15, pp. 127-146.

Leonard B P (1980)
The QUICK Algorithm: A Uniformly Third-Order Finite-Difference Method for Highly Convective Flow
Computer Methods in Fluids
Editors: Morgan K, Taylor C and Brebbia C A, Pentech Press, London, 1980, pp.159.

Leonard B P (1988)
Universal Limiter for Transient Interpolation Modeling of the Advective Transport Equations: The ULTIMATE Conservative Difference Scheme
NASA Tech. Memo. 100916, ICOMP-88-11, Lewis Research Center, Cleveland, OH, USA, 1988.

Lewis R W, Usmani A S and Huang H C (1991a)
Casting Modelling by Finite Elements - Our Experience
Modeling of Casting, Welding and Advanced Solidification Processes V
Editors: Rappaz M, Ozgu M R and Mahin K W, Publisher: The Minerals, Metals and Materials Society, 1991, pp. 3-14.

Lewis R W, Usmani A S and Cross J T (1991b)
Finite Element Modelling of Mould Filling
Finite Elements in the 90's
Editors: Onate E, Periaux J and Samuelsson A, publisher: Springer-Verlag/CIMNE, Bacelona, 1991, pp. 441-449.

Lin H J and Hwang W S (1988)
Combined Fluid Flow and Heat Transfer Analysis for the Filling of Castings
American Foundrymen's Society Transactions, 1988, vol. 144, pp. 447-458.

Lin H J and Hwang W S (1989)
Three-Dimensional Fluid Flow Simulation for Mold Filling
American Foundrymen's Society Transactions, 1989, vol. 171, pp 855-862.

Liu Jun (1985)
Application of "Variable Blockage" Method to Steady Flows over Curved Beds
CFDU Report CFD/85/12, Imperial College, London, 1985.

Liu Jun (1986a)
The Scalar-Equation Method for the Prediction of Free-Surface Flows
PDR/CFDU IC/25, CFDU, Mech. Eng. Dept., Imperial College, London UK, March, 1986.

Liu Jun (1986b)
Computer Modelling of Flows with a Free Surface
PhD Thesis, CFDU, Mech. Eng. Dept., Imperial College, London, UK, October, 1986.

Liu Jun and Spalding D B (1988)
Numerical Simulation of Flows with Moving Interfaces
PCH PhysicoChemical Hydrodynamics, 1988, vol. 10, No. 5/6, pp. 625-637.

MAGMA
(MAGMAsoft - a CFD based casting simulation software)
MAGMA Giessereitechnologie GMBH, Adalbertstrasse 116-118, D5100 Aschen, Germany

Markatos N C and Pericleous K A (1984)
Laminar and Turbulent Natural Convection in an Enclosed Cavity
Int. J. Heat Mass Transfer, 1984, vol. 27, pp, 755-772.

Maxwell T T (1977)
Numerical Modelling of Free-Surface Flows
PhD Thesis, Mech. Eng. Dept., Imperial College, London, UK, 1977.

Medina D E, Liggett J A, Birchwood R A and Torrance K E (1991)
A Consistent Boundary Element Method for Free Surface Hydrodynamic Calculations
Int. J. for Numerical Meth. in Fluids, 1991, vol. 12, pp. 835-857.

Melaaen M C (1992a)
Calculation of Fluid Flows with Staggered and Nonstaggered Curvilinear Nonorthogonal Grids - The Theory
Numerical Heat Transfer, Part B, 1992, vol. 21, pp 1-19.

Melaaen M C (1992b)
Calculation of Fluid Flows with Staggered and Nonstaggered Curvilinear Nonorthogonal Grids - A Comparison
Numerical Heat Transfer, Part B, 1992, vol. 21, pp 21-39.

Mishima S and Szekely J (1989)
The Modeling of Fluid Flow and Heat Transfer in Mold Filling
ISIJ International, 1989, vol. 29, no. 4, pp. 324-332.

Miyata H and Nishimura S (1985)
Finite-Difference Simulation of Nonlinear Ship Waves
Journal of Fluid Mechanics, 1985, vol. 157, pp. 327-357.

Modeling of Casting, Welding and Advanced Solidification Processes VI
Editors: Piwonka T S, Voller V and Katgerman L, Publisher: The Minerals, Metals & Materials Society, Pennsylvania, USA, 1993.

Nichols and Hirt C W (1971)
Improved Free Surface Boundary Conditions for Numerical Incompressible-Flow Calculations
J. Comp. Phys., 1971, vol. 8, pp. 433-448.

Nichols B D and Hirt C W (1973)
J. Comput. Physics, 1973, vol. 12, pp. 234.

Patom I S (1987)
Application of the VOF Method to the Sloshing of a Fluid in a partially filled Cylindrical Container
Int. J. for Numerical Meth. in Fluids, 1987, vol. 7. pp. 535-550.

Patankar S V (1980)
Numerical Heat Transfer and Fluid Flow
Hemisphere Pub. Corp., Washington DC, 1980.

Patankar S V and Spalding D B (1972)
A Calculation Procedure for Heat, Mass and Momentum Transfer in Three-Dimensional Parabolic Flows
Int. Journal of Numerical Heat and Mass Transfer, 1972, vol. 5, pp. 1787-1806.

Pericleous K A and Rhodes N (1984)
The Modelling of a Cyclonic Floatation Machine
CHAM Report 3251, CHAM Limited, Wimbledon, London, UK, 1984.

Pericleous K A and Drake S N (1985)
An Algebraic Slip Model of PHOENICS for Multi-phase Applications
Proceedings Numerical Simulation of Fluid Flow Heat/Mass Transfer Processes
Publisher: Springer-Verlag, 1985, pp. 18.

Pericleous K A, Chan A (K S), and Cross M (1993)
Three-Dimensional Simulation of the Filling of Moulds by Liquid Metals
Mathematical Modelling for Material Processing
Editors: Cross M, Pittman J and Wood R, 1993, pp. 273-281.

Pericleous K A and Chan K S (1993)
The SEA method for Free surface problems with Heat Transfer and Change of Phase
Numerical Methods in Multiphase Flows 1994
Editors: Crowe C T, Johnson R, Prosperetti A, Sommerfeld M and Tsuji Y,
Publisher: American Society of Mechanical Engineers, 1994, FED-Vol. 185, pp. 227-236.

PHOENICS Beginner's Guide (1987)
by Rosten H I and Spalding D B
CHAM TR/100, CHAM Limited, Wimbledon, London, UK, 1987.

PHOENICS Input Library
by Rosten H I and Spalding D B
CHAM TR/101, CHAM Limited, Wimbledon, London, UK.

Portland Group (1992)
<u>PGTools Documentation</u>, The Portland Group, Wilsonville, Oregon, USA, 1992.

Polmear I (1991)
<u>Metallurgy of the Light Alloys</u>
Publisher: Butterworth-Heinemann Ltd, 1991, UK.

Preddy K (1992)
Private communication; Stone Foundry, Woolwich, London, 1992.

Preddy K (1993)
Private communication; Stone Foundry, Woolwich, London, 1993.

ProCAST User Manual (1992)
<u>ProCAST User Manual</u>, version 2.1, UES Inc., Dayton, Ohio, USA, 1992.

Ramshaw J D and Trapp J A (1976)
A Numerical Technique for Low Speed Homogeneous Two-Phase with Sharp Interfaces
Journal of Computational Physics, 1976, vol. 21, pp. 428-452.

Rhie C M and Chow W L (1982)
A Numerical Study of the Turbulent Flow Past an Isolated Airfoil with Trailing Edge Separation
AIAA J., 1982, vol. 21, pp. 1525-1532.

Samonds M and Waite D (1993)
3D Finite Element Simulation of Filling Transients in Metal Castings
<u>Mathematical Modelling for Material Processing</u>
Editors: Cross M, Pittman J and Wood R, 1993, pp. 283-300.

Shen S F (1992)
Fixed-Domain Approach to the Cavity-Filling Problem
Forming Processes Conference - NUMIFORM 92, France, 1992.

Simitovic R Renell M, Ludwig J and Pericleous K (1986)
A 3D Model of Propagation, Dissipation and Reflection of Giant Waves generated by Landslide Impact into Dam Lakes
CHAM Report, CHAM Limited, Wimbledon, UK, 1986.

Smith T G and Wilke J O (1975)
Laminar Free-Surface Flow into a Vertical Cylinder
Computers and Fluids, 1975, vol. 3, pp. 51-68.

Spalding D B (1972)
A Novel Finite-Difference Formulation for Differential Expressions Involving Both First and Second Derivatives
Int. J. Num. Methods Eng., 1972, vol. 4, pp. 551.

Spalding D B (1974)
A Method for Computing Steady and Unsteady Flows Possessing Discontinuities of Density
CHAM Report 910/2, CHAM Limited, Wimbledon, London, UK, 1974.

Spalding D B (1977)
The Calculation of Free-Convection Phenomena in Gas-Liquid Mixtures
Heat Transfer and Turbulent Buoyant Convection Volume II,
Editors: Spalding D and Afgan N, Hemisphere Pub. Corp., Washington DC, USA, 1977.

Spalding D B (1980a)
Numerical Computation of Multi-Phase Fluid Flow and Heat Transfer
Recent Advance In Numerical Methods In Fluids Volume 1,
Editors: Taylor C and Morgan K, Pineridge Press, Swansea, UK, 1980.

Spaling D B (1980b)
Mathematical Modelling of fluid-mechanics, heat-transfer and chemical-reaction processes
A Lecture Course, CFDU Report HTS/80/1, Imperial College, London, UK, 1980.

Stone N L (1968)
Iterative Solution of Implicit Approximation of Multidimensional Partial Differential Equations
SIAM Journal of Numerical Analysis, 1968, vol. 5, pp. 530-558.

Swaminathan C R and Voller V R (1993)
An 'Enthalpy Type' Formulation for the Numerical Modeling of Mold Filling
Modeling of Casting, Welding and Advanced Solidification Processes VI
Editors: Piwonka T S, Voller V and Katgerman L, Publisher: The Minerals, Metals & Materials Society, Pennsylvania, USA, 1993, pp. 365-372.

Tomiyama A and Sou A (1991)
Numerical Simulation of a Single Bubble Rising in Liquids using the Volume of Fluid Method
Proceeding of the International Conference on Multiphase Flows '91
Tsukuba, Japan, 1991, pp. 373-376.

Transtech TTM110 (1991)
i860 based Transputer Module
Transtech Parallel Systems Ltd, High Wycombe, UK, 1991.

UES Inc.
(ProCAST - CFD based casting simulation software)
UES Inc., Dayton, Ohio, USA.

Unverdi S O and Tryggvason G (1992)
A Front-Tracking Method for Viscous, Incompressible, Multi-fluid Flows
Journal of Computational Physics, 1992, vol. 100, pp. 25-37.

Usmani A S, Cross J T and Lewis R W (1992)
A finite element model for the simulation of mould filling in metal casting and the associated heat transfer
Report CR/698/92, Civil Engineering, Univ College of Swansea, Wales, 1992.

Van Doormall J P and Raithby G D (1984)
Enhancements of the SIMPLE Method for Predicting Incompressible Fluid Flows
Numerical Heat Transfer, 1984, vol. 7, pp. 147-163.

Van Leer B (1977)
Towards the Ultimate Conservative Difference Scheme. IV. A New Approach to Numerical Convection
Journal of Computational Physics, 1977, vol. 23, pp. 276-299.

Villasenor F (1988)
Numerical simulation of Kelvin-Helmholtz instability in a stratified shear flow
PhD, CFDU, Imperial College, 1988 and also at PCH, 1987, vol. 9, No. 1/2, pp 379-386.

Voller V R, Markatos N C, and Cross M (1985)
Techniques for Accounting for the Moving Interface in Convection/Diffusion Phase Change
Numerical Methods in Thermal Problems Vol. 4
Editors: Lewis R W and Morgan K, Pineridge Press, Swansea, UK, 1985, pp 595-609.

Voller V R, Cross M, and Markatos N C (1987a)
An Enthalpy Method for Convection/Diffusion Phase Changes
Int. J. Num. Meth. Engng, 1987, vol. 24, pp. 271-284.

Voller V R and Prakash C (1987b)
A Fixed Grid Numerical Modelling Methodology for Convection-Diffusion Mushy Region Phase-Change Problems
Int. J. Heat Transfer, 1987, vol. 30, pp. 1709-1719.

Voller V R and C R Swaminathan (1991)
General Source-Based Method for Solidification Phase Change
Numerical Heat Transfer, Part B, 1991, vol. 19, pp. 175-189.

Welch J E, Harlow F H, Shannon J P and Dally B J (1966)
The MAC Method: A Computing technique for Solving Viscous incompressible Transient Fluid Flow Problems Involving Free Surfaces
Report LA-3425, Los Alamos Scientific Laboratory, Los Alamos, NM, USA, 1966.

Yeung R W (1982)
Numerical Methods in Free-Surface Flows
Ann. Rev. Fluid Mech., 1992, vol. 12, pp. 395-442.

Youngs D L (1982)
Time-Dependent Multi-Material Flow with Large Fluid Distortion
Numerical Methods for Fluid Dynamics
Editors: Morton K and Baines M, Academic Press, 1982, pp. 273-285.

Zhang Y F, Liu W K and Wang H P (1993)
Casting filling simulations of thin-walled cavities with solidification
Modelling of Casting, Welding and Advanced Solidification Processes-VI,
Editors: Piwonka T S, Voller V and Katgerman L, Publisher: The Minerals, Metals & Materials Society, Pennsylvania, USA, 1993, pp 413-420.

Appendix A *COLLAPSE OF A WATER COLUMN*

A1.0 Collapse of a Water Column

This appendix is intended to provide information on a problem which the author had used to test his 2-D implementation of the SEA algorithm. The results are compared with those obtained by Liu Jun (1986b) and Hirt and Nichols (1981). They are shown in later sections.

A1.1 Problem Specification

The configuration of the problem is depicted in *Figure A1.1*. A rectangular column of water of dimension 1 m (wide) by 2 m (high) is initially confined between two vertical walls in hydrostatic equilibrium. At the beginning of the simulation, one wall is removed causing the water column to collapse under its own weight due to the downward gravitational force. This causes the water to flow across the flat surface from left to right. Eventually, the water reaches its lowest possible level and once again remain in hydrostatic equilibrium.

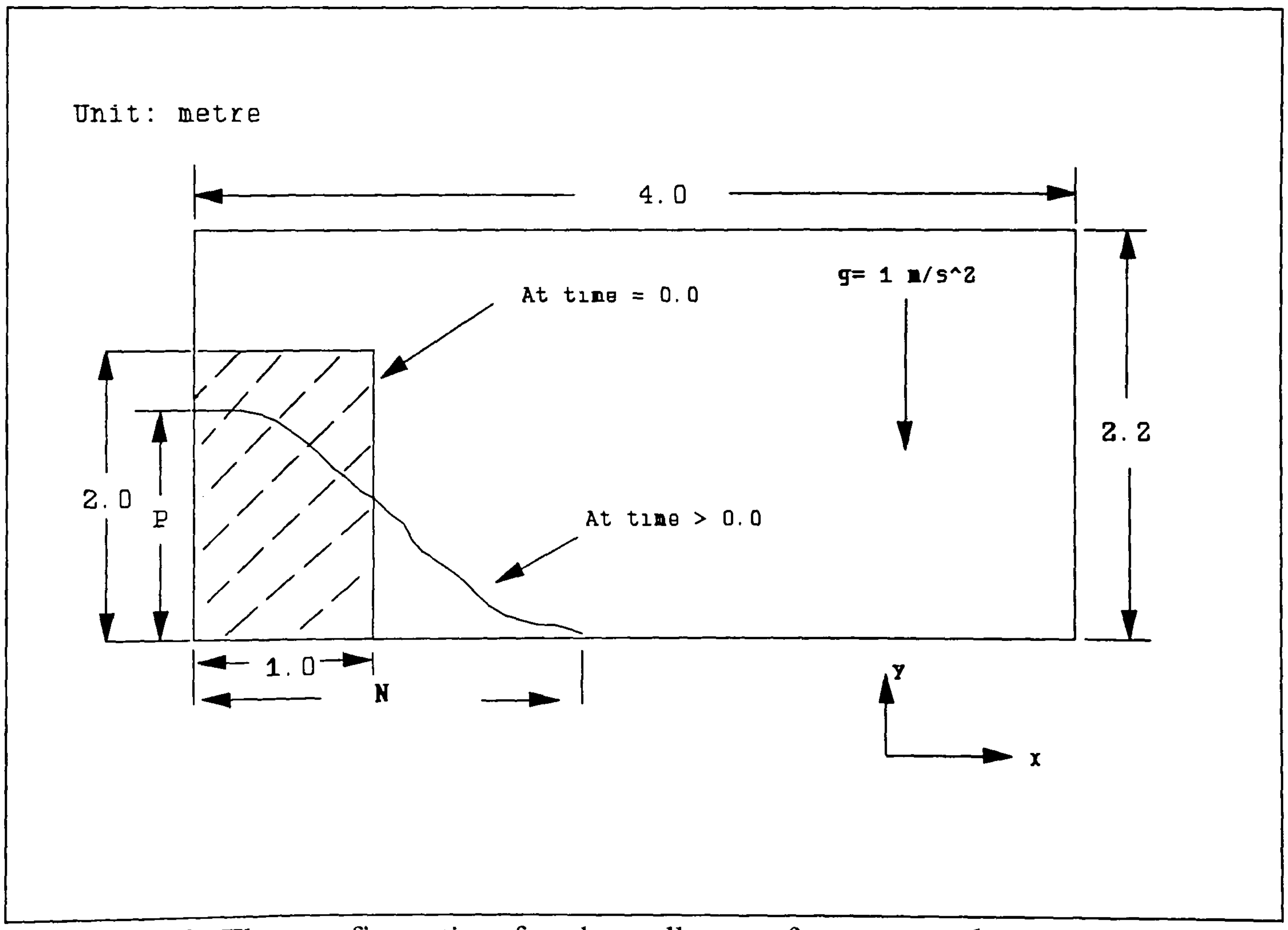


Figure A1.1 The configuration for the collapse of a water column case.

The dimension of the computational domain is 4 metres long by 2.2 metres high, and the domain is divided uniformly into a total of 40 x 22 = 880 cells. The fluid properties of the air and water are listed in *Table A1.1.* The initial and boundary conditions, and the run time parameters used are listed in *Tables A1.2* and *A1.3* respectively.

	Water	Air
Density, ρ (Kg/m^3)	998.0	1.205
Kinematic Viscosity, ν (m^2/s)	1.012x10^{-6}	1.5x10^{-5}

Table A1.1 Fluid properties.

A1.2 Results and Remarks

Figure A1.2 shows a side-by-side comparison of the free surface profiles of the water column problem predicted by Hirt and Nichols' VOF method, the author's implementation of Liu Jun's method, and those reported by Liu Jun. All the free surface profiles are plotted at $\phi = 0.5$. It can clearly be seen that the free surface profiles predicted by the author and Liu Jun's implementations of the scalar equation method are in close agreement with those reported by Hirt and Nichols. Some discrepancies between Hirt and Nichols' and those of the author and Liu Jun's can be observed at the leading edge of the water front. This is mainly due to larger number of cells used along the bottom wall in the Hirt and Nichols' simulation.

Figure A1.3 shows a comparison of velocity vector plots of the same problem between those reported by Liu Jun and the results predicted by the author's implementation. The trends and profiles of these plots are in close agreements.

Initial conditions:

Initialize the water column: At time equals zero, the first 10 x 20 cells from the bottom left of the domain have their ϕ values set to 1 to represent water.

Reference Pressure: A zero pressure is set to a cell at the top right corner of the domain.

Boundary conditions:

Walls: impermeable, constant temperature, and non-slip or zero velocity at the four walls is assumed.

Gravity acting downward: 1.0 ms^{-2}

Table A1.2 Initial and boundary conditions.

Simulation of real time: 2.0 s

Time-step size: 10^{-2} s

Total number of time-steps: 200

Number of sweep per time-step: 60

Residual tolerance for velocities and pressure : 10^{-8}

Relaxation factors:

Linear: pressure (0.8)

False time-step: u (0.05), v (0.01)

Table A1.3 Run-time parameters.

Comparisons of the height and front (leading edge) locations of the water column predicted by the author's and Liu Jun's implementations can be found in *Figures A1.4* and *A1.5* respectively.

The locations of the water column height and the front position are plotted using the following dimensionless variables:

$$\boldsymbol{L} = \frac{N}{a} \;;\; \boldsymbol{H} = \frac{P}{b} \;;\; \boldsymbol{T} = t\sqrt{\left(\frac{g}{a}\right)}$$

where N and P represent the leading edge and height locations (as shown in *Figure A1.1*). t is the time (in seconds) and g is the gravitational acceleration which is set to 1 ms^{-2}. And finally, a and b are set to 1 and 2 m respectively. These locations are calculated using interpolation at ϕ=0.5 between cell centres.

These figures show that the predictions produced by the author's implementation are closely matched of those reported by Liu Jun. However, some discrepancies are noted when T is greater than 1.2 in both figures. These may be due to slight numerical smearing at the air-water interface, especially at the front leading edge of the water.

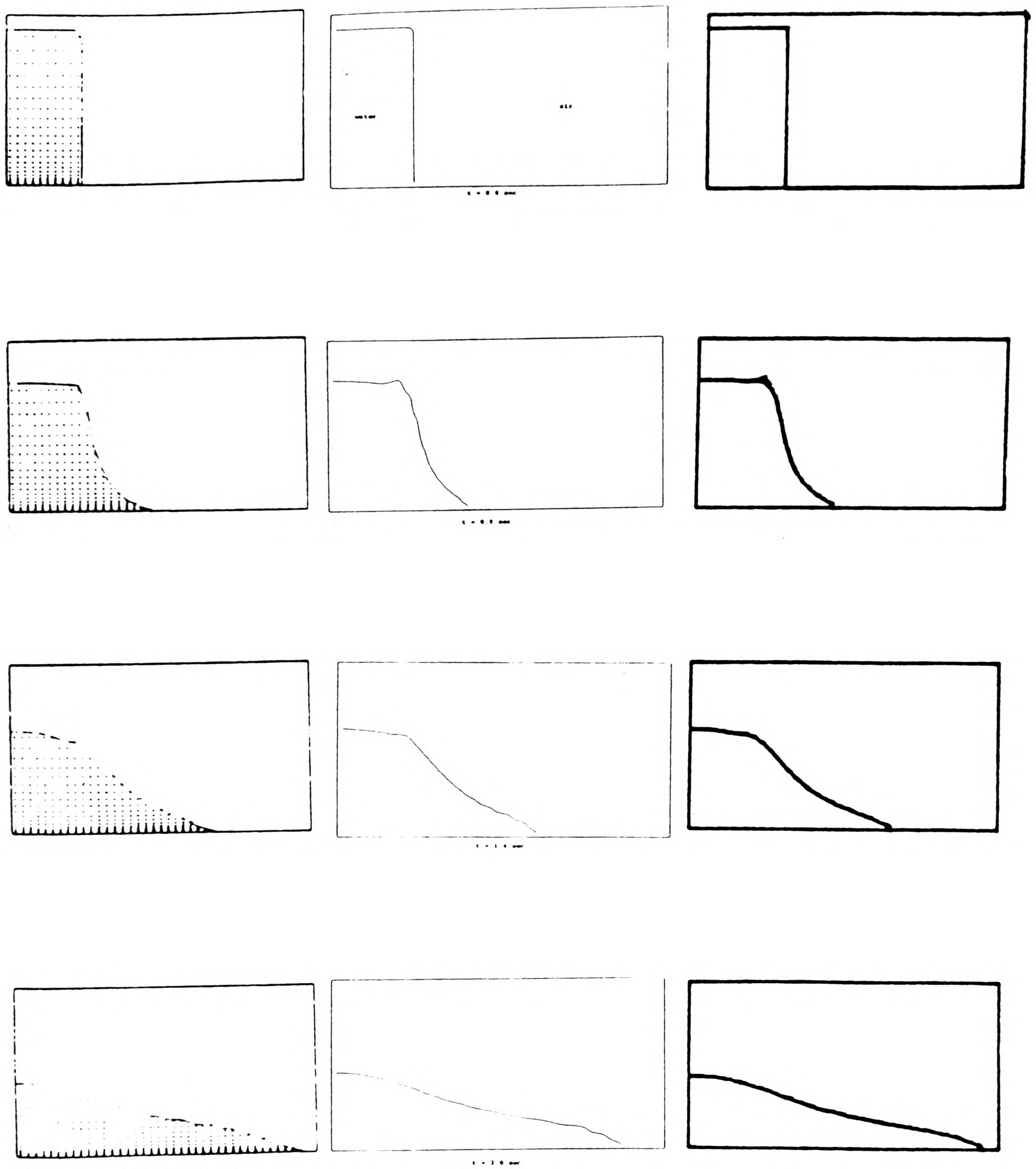


Figure A1.2 Free surface profiles for the collapse of water column problem at time = 0.0, 0.9, 1.4 and 2.0 seconds.

First column: Hirt and Nichols' VOF method.
Second column: Present study.
Third column: Liu Jun's 2-D scalar equation method.

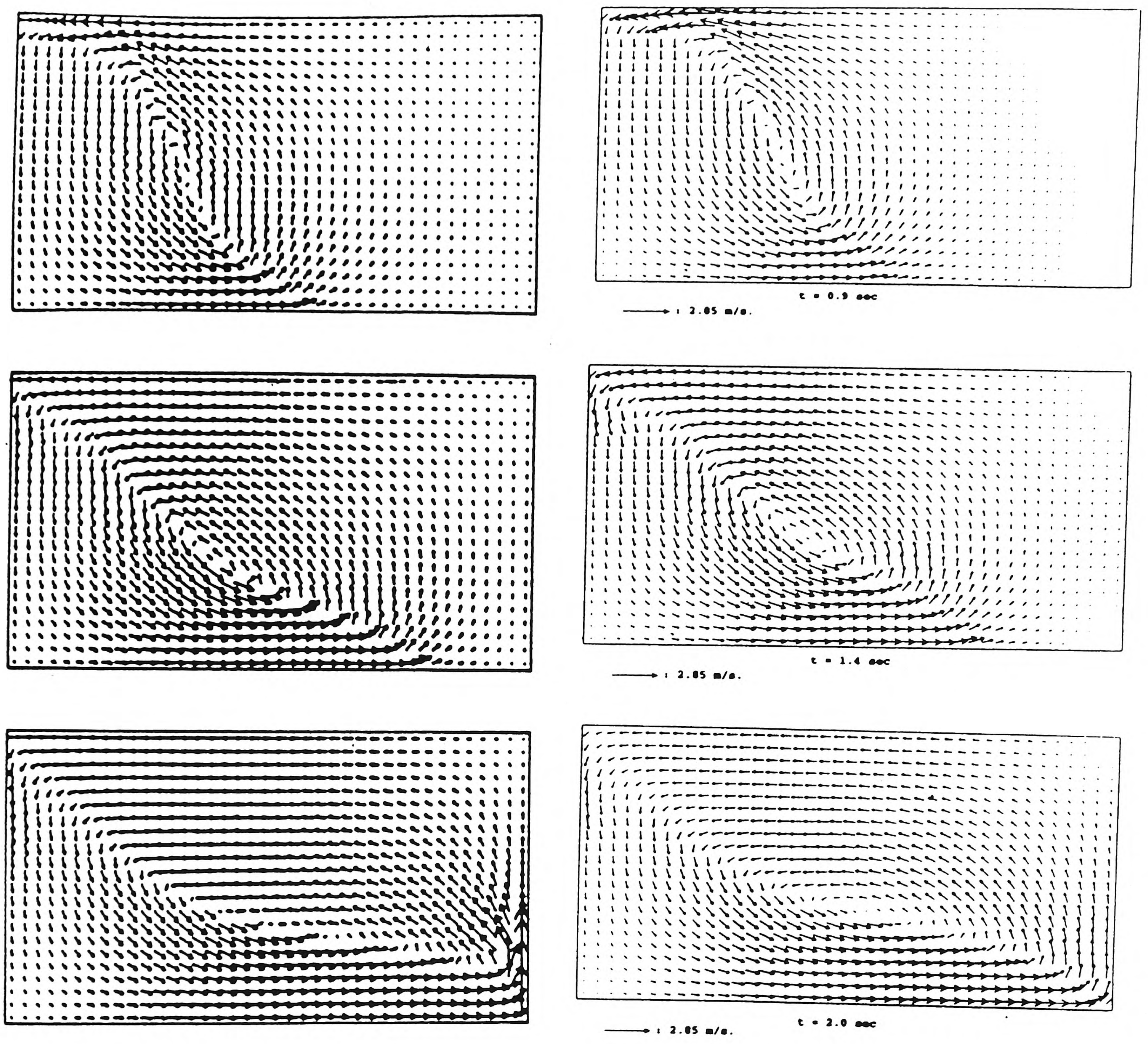


Figure A1.3 Velocity vector plots for the collapse of water column problem at time = 0.9, 1.4 and 2.0 seconds.

Left column: Liu Jun's 2-D scalar equation method.
Right column: Present study.

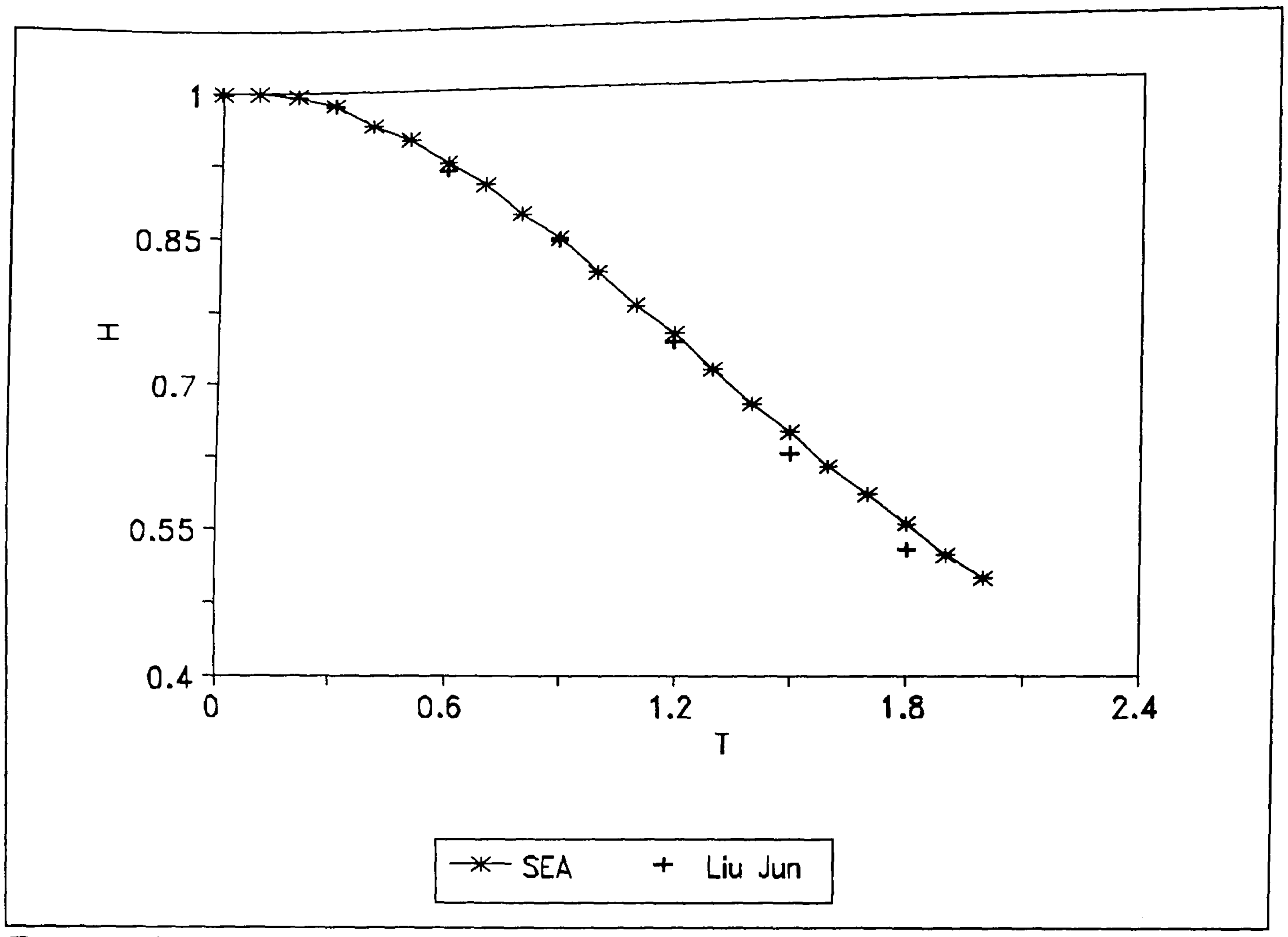


Figure A1.4 Comparison of wave height locations - Present study vs Liu Jun's data.

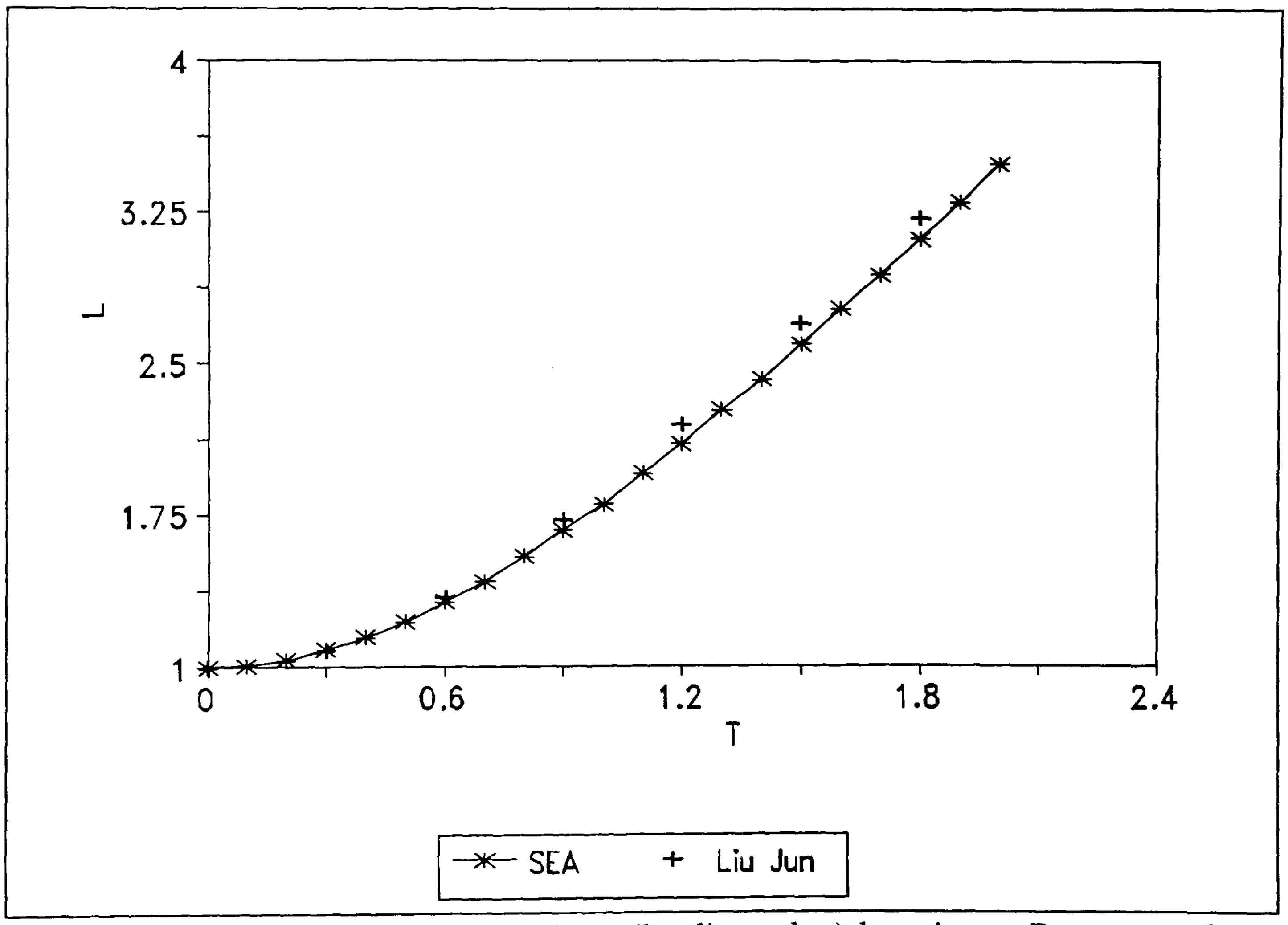


Figure A1.5 Comparison of wave front (leading edge) locations - Present study vs Liu Jun's data.