A Two Dimensional Algorithm for the Non-linear Equations of Gas Dynamics Employing Operator Splitting

J.J. BARLEY

Numerical Analysis Report No. 4/87

The work reported forms part of the Research Programme of the Oxford/Reading Institute for Computational Fluid Dynamics.

Abstract

An approximate (linearised) Riemann solver for the solution of the Euler Equations in two dimensions incorporating operator splitting is applied to two test problems, an infinite spherically divergent shock and a bursting membrane problem.

Contents

1. Introduction

Prompted by the work of Roe and Pike [4] and of Glaister [5], we study the linearised approximate Riemannn solver of Roe [6] for the solution of the one-dimensional Euler equations of gas dynamics.

In this report the method is used to investigate the technique of operator splitting in the solution of the two-dimensional Euler equations by considering two test problems, those of an infinite spherically diverging shock and of a bursting cylindrical membrane.

In section 2 we state the Euler equations for an ideal gas and in section 3 we consider the Jacobians, eigenvalues and eigenvectors of the flux functions for these equations. In section 4 we briefly outline the technique of operator splitting and in section 5 describe the linearised approximate Riemann solver. The two test problems are introduced in section 6 and the methods of section 5 are used to produce numerical results which are shown in section 7.

Some discussion of the results is given in section 8 and a note on programming is offered in an Appendix.

2. Statement of the Equations

In this section we state the equations that govern the two dimensional motion of an inviscid compressible fluid and write them as a first order system of hyperbolic conservation laws.

These three equations, written as conservation laws, are

(i) conservation of mass

$$\frac{\partial \rho}{\partial t} + \underset{\sim}{\nabla} \cdot (\rho \underset{\sim}{u}) = 0 \tag{2.1}$$

(ii) conservation of momentum

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

(iii) conservation of energy

$$\frac{\partial e}{\partial t} + \underset{\sim}{\nabla} \cdot (\underset{\sim}{u}(e + p)) = 0 \tag{2.3}$$

where the (conserved) variables are density, ρ, momentum, $\rho \underset{\sim}{u}$ (or $\underset{\sim}{m}$) and energy e.

$\rho = \rho(\underset{\sim}{x},t)$, $\underset{\sim}{u} = \underset{\sim}{u}(\underset{\sim}{x},t) = (u(\underset{\sim}{x},t)\ ,\ v(\underset{\sim}{x},t))^T$,

$p = p(\underset{\sim}{x},t)$, $i = i(\underset{\sim}{x},t)$, $e = e(\underset{\sim}{x},t)$ represent density, velocity (in two co-ordinate directions), pressure, specific internal energy and total energy respectively at a general position $\underset{\sim}{x} = (x,y)$ at time t. The three conservation laws, together with an equation of state

$$p = p(\rho, i) \tag{2.4}$$

constitute the Euler equations of compressible flow.

For an ideal gas, the equation of state (2.4) is

$$p = (\gamma-1)\rho i \qquad (2.5)$$

where γ is the gas constant for the particular gas we are considering, e.g. $\gamma = 1.4$ for air. Total energy is related to specific internal energy by the relationship

$$e = \rho i + \tfrac{1}{2}\rho q^2 , \qquad (2.6)$$

$$q^2 = u^2 + v^2 . \qquad (2.7)$$

We can write these equations as a single system by putting

$$\underset{\sim}{u} = (\rho, \rho u, \rho v, e)^T \qquad (2.8)$$

$$\underset{\sim}{F}(\underset{\sim}{u}) = (\rho u, p+\rho u^2, \rho uv, u(e+p))^T \qquad (2.9)$$

$$\underset{\sim}{G}(\underset{\sim}{u}) = (\rho v, \rho uv, p+\rho v^2, v(e+p))^T \qquad (2.10)$$

Then the conservation laws (2.1), (2.2), (2.3) can be written in the compact form

$$\underset{\sim}{u}_t + \underset{\sim}{F}(\underset{\sim}{u})_x + \underset{\sim}{G}(\underset{\sim}{u})_y = 0 \qquad (2.11)$$

We now have a first order system of hyperbolic conservation laws.

Equation (2.11) together with the equation of state for an ideal gas (2.5) constitute the Euler equations in two dimensions.

For development of a Riemann solver using a general equation of state, see e.g. Glaister [5].

3. Jacobians, Eigenvalues and Eigenvectors

Taking the system of hyperbolic conservation laws

$$\underset{\sim}{u}_t + \underset{\sim}{F}(\underset{\sim}{u})_x + \underset{\sim}{G}(\underset{\sim}{u})_y = 0 \qquad (3.1)$$

we can write this as

$$\underset{\sim}{u}_t + A(\underset{\sim}{u})\, \underset{\sim}{u}_x + B(\underset{\sim}{u})\, \underset{\sim}{u}_y = 0 \qquad (3.2)$$

where $A(\underset{\sim}{u})$ and $B(\underset{\sim}{u})$ are the Jacobians of $F(\underset{\sim}{u})$ and $G(\underset{\sim}{u})$ respectively,

i.e.

$$A(\underset{\sim}{u}) = \frac{\partial \underset{\sim}{F}}{\partial \underset{\sim}{u}} \qquad (3.3)$$

$$B(\underset{\sim}{u}) = \frac{\partial \underset{\sim}{G}}{\partial \underset{\sim}{u}} \qquad (3.4)$$

each of which has real eigenvalues.

We consider the problem of finding the eigenvalues and right eigenvectors of the two Jacobian matrices, A and B, since this will form the basis of the Riemann solver.

Writing the momentum $\underset{\sim}{m} = \rho \underset{\sim}{u}$

as $\underset{\sim}{m} = (m,n)^T$

some simple algebra reveals the Jacobian A to be

$$A(\underset{\sim}{u}) = (\underset{\sim}{A}_1, \underset{\sim}{A}_2, \underset{\sim}{A}_3, \underset{\sim}{A}_4) \qquad (3.5)$$

where

$$\underset{\sim}{A}_1^T = \left[0,\ \frac{(\gamma-1)n^2}{2\rho^2} - \frac{(3-\gamma)m^2}{2\rho^2}\ ,\ -\frac{nm}{\rho}\ ,\ -\frac{m\gamma e}{\rho^2} + \frac{m(\gamma-1)(m^2+n^2)}{\rho^3} - \frac{\gamma em}{\rho^2}\right]$$

$$\underset{\sim}{A}_2^T = \left[1,\ \frac{(3-\gamma)m}{\rho}\ ,\ \frac{n}{\rho}\ ,\ \frac{\gamma e}{\rho} - \frac{(\gamma-1)}{2\rho^2}(3m^2 + n^2) + \frac{\gamma e}{\rho}\right]$$

$$\underset{\sim}{A}_3^T = \left[0, -\frac{(\gamma-1)n}{\rho}\ ,\ \frac{m}{\rho}\ ,\ -\frac{mn(\gamma-1)}{\rho^2}\right]$$

$$\underset{\sim}{A}_4^T = \left[0,\ (\gamma-1)\ ,\ 0\ ,\ \frac{\gamma m}{\rho}\right]$$

with a similar form for $B(\underset{\sim}{u})$.

The calculation of the eigenvalues and eigenvectors is straightforward and is indicated by Roe [6] and Glaister [5]. We consider $A(\underset{\sim}{u})$ now and $B(\underset{\sim}{u})$ later. Calculation yields the eigenvalues of A to be

$$\lambda_1 = u - a \tag{3.6a}$$

$$\lambda_2 = u \tag{3.6b}$$

$$\lambda_3 = u \tag{3.6c}$$

$$\lambda_4 = u + a \tag{3.6d}$$

where a is the sound speed given by

$$a^2 = (\gamma-1)\ (H-\tfrac{1}{2}q^2) \tag{3.7}$$

H is the enthalpy defined by

$$H = \frac{e + p}{\rho} \tag{3.8}$$

and q is the fluid speed given earlier. The corresponding right eigenvectors are

$$\underset{\sim}{e}_1 = \begin{bmatrix} 1 \\ u-a \\ v \\ H-ua \end{bmatrix} \qquad \underset{\sim}{e}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ v \end{bmatrix}$$

$$\underset{\sim}{e}_3 = \begin{bmatrix} 1 \\ u \\ v \\ \frac{1}{2}q^2 \end{bmatrix} \qquad \underset{\sim}{e}_4 = \begin{bmatrix} 1 \\ u+a \\ v \\ H+ua \end{bmatrix}$$

Analysis of the Jacobian $B(\underset{\sim}{u})$ reveals that it has eigenvalues

$$\lambda_1 = v - a$$

$$\lambda_2 = v$$

$$\lambda_3 = v$$

$$\lambda_4 = v + a$$

with corresponding eigenvectors

$$\underset{\sim}{e}_1 = \begin{bmatrix} 1 \\ u \\ v-a \\ H-va \end{bmatrix} \qquad \underset{\sim}{e}_2 = \begin{bmatrix} 1 \\ u \\ v \\ \frac{1}{2}q^2 \end{bmatrix}$$

$$\underset{\sim}{e}_3 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ u \end{bmatrix} \qquad \underset{\sim}{e}_4 = \begin{bmatrix} 1 \\ u \\ v+a \\ H+va \end{bmatrix}$$

In section 5 it will be shown how these eigenvalues and eigenvectors form the basis of the Riemannn solver.

4. Operator Splitting

The technique used to solve the two dimensional test problems described later is that of operator splitting, which we now outline.

Consider the two dimensional linear advection equation

$$u_t + au_x + bu_y = 0 \tag{4.1}$$

We study the splitting of (4.1) into two one dimensional advection equations (see Yanenko [7], Strang [14])

$$\tfrac{1}{2}\, u_t + au_x = 0 \tag{4.2}$$

$$\tfrac{1}{2}\, u_t + bu_y = 0 \tag{4.3}$$

If L_x is a numerical solution operator of (4.2) and L_y is a numerical solution operator of (4.3), there are several options on how we may combine L_x , L_y to solve (4.1) and retain the accuracy of the underlying one-dimensional scheme.

Consider the system of equations

$$\underset{\sim}{u}_t + A\underset{\sim}{u}_x + B\underset{\sim}{u}_y = 0 \tag{4.4}$$

which we again split into two one dimensional equations

$$\tfrac{1}{2}\, \underset{\sim}{u}_t + A\underset{\sim}{u}_x = 0 \tag{4.5}$$

$$\tfrac{1}{2}\, \underset{\sim}{u}_t + B\underset{\sim}{u}_y = 0 \tag{4.6}$$

Again, let L_x and L_y be solution operators of (4.5) and (4.6), respectively, both of order p say. Sod [8] has shown that the order of the split scheme is affected by the order in which we apply solution operators L_x and L_y . For example, if

A and B commute then applying the solution operators in a straightforward manner i.e. $u^{n+1} = L_x L_y (u^n)$ will produce a solution which is also of order p (p = 1, 2, ...), but if A and B do not commute then the above solution will be at most first order accurate! However, second order accuracy can be achieved in two ways, firstly by computing u^{n+1} using

$$u^{n+1} = \tfrac{1}{2}(L_x L_y + L_y L_x)(u^n) \tag{4.7}$$

which is an averaging process, or secondly by advancing the solution from $n\Delta t$ to $(n+1)\Delta t$ in four quarter steps

$$u^{n+1} = L_x^{\frac{1}{2}} L_y^{\frac{1}{2}} L_y^{\frac{1}{2}} L_x^{\frac{1}{2}} (u^n) \tag{4.8}$$

where the superscript denotes the fraction of Δt used in that solution operator. It is shown in Sod [8] that

$$L_x^{\frac{1}{2}} L_y^{\frac{1}{2}} L_y^{\frac{1}{2}} L_x^{\frac{1}{2}} \equiv L_x^{\frac{1}{2}} L_y L_x^{\frac{1}{2}} . \tag{4.9}$$

Thus, a consistent numerical algorithm may be constructed as follows:-

1. Apply the solution operator L_x with timestep $\Delta t/2$ along each line y = constant for all such lines, thus solving (4.2) on the whole numerical grid. Update the solution. (This constitutes one X-sweep).

2. Apply the solution operator L_y with time step Δt along each line x = constant for all such lines, thus solving (4.3) on the whole numerical grid. Update the solution. (This constitutes a Y-sweep).

3. Repeat step 1.

   After the final update, we have completed one time step.

In effect we are not solving the two dimensional problem in a genuinely two dimensional manner, rather, we are solving the problem in a one-dimensional manner sequentially in the x and y co-ordinate directions.

5. An Approximate Riemann Solver

Following in the footsteps of Roe [4], [6] and Glaister [5], we describe the essential points of Roe's approximate linearised Riemann solver in two dimensions for the Euler euqations incorporating the technique of operator splitting.

We first consider solving

$$\underset{\sim}{u}_t + \underset{\sim}{F}(\underset{\sim}{u})_x = 0 \tag{5.1}$$

along a data line y = constant. Equation (5.1) can be written as

$$\underset{\sim}{u}_t + A(\underset{\sim}{u})\, \underset{\sim}{u}_x = 0 \ . \tag{5.2}$$

We construct an approximation to the solution of the equation (5.1) with piecewise constant data by solving a set of Riemann problems. To do this in the manner proposed by Roe we assume that A can be linearised in $\underset{\sim}{u}_L$, $\underset{\sim}{u}_R$ (the values of $\underset{\sim}{u}$ at the left and right hand ends of the computational cell) such that A is constant within the computational cell $(x_L \ , \ x_R)$. Since our data is only provided pointwise, we have values of the variables only at the left and right hand ends of each cell. Thus, we need to consider an approximation to A , denoted by $\tilde{A}$ say, that is some average of values of A at the ends of the cell. Roe shows that $\tilde{A}$ must satisfy the following properties

$$\tilde{A} = \tilde{A}(\underset{\sim}{u}_L, \underset{\sim}{u}_R) \quad \text{s.t.}$$

1) $\tilde{A}$ constitutes a linear mapping from $\underset{\sim}{u}$ to $\underset{\sim}{f}$

2) $\underset{\sim}{u}_L \rightarrow \underset{\sim}{u} \leftarrow \underset{\sim}{u}_R$, $\tilde{A}(\underset{\sim}{u}_L, \underset{\sim}{u}_R) \rightarrow A(\underset{\sim}{u})$ (5.3)

3) For any $\underset{\sim}{u}_L, \underset{\sim}{u}_R$

$$\tilde{A}(\underset{\sim}{u}_L, \underset{\sim}{u}_R) \times (\underset{\sim}{u}_L - \underset{\sim}{u}_R) = \underset{\sim}{F}_L - \underset{\sim}{F}_R \tag{5.4}$$

since $\tilde{A}\underset{\sim}{u} = \underset{\sim}{F} \Rightarrow \underset{\sim}{F}_x = \tilde{A}\underset{\sim}{u}_x$

4) Eigenvectors of $\tilde{A}$ must be linearly independent.

These four conditions are necessary and sufficient for the algorithm to recognise a shockwave and also for the algorithm to be conservative.

We note a result due also to Roe. If $(\underset{\sim}{u}_L, \underset{\sim}{u}_R)$ satisfy the Rankine-Hugoniot jump relationship

$$\underset{\sim}{F}_L - \underset{\sim}{F}_R = S(\underset{\sim}{u}_L - \underset{\sim}{u}_R) \tag{5.6}$$

for some scalar S, the shock speed, then S is an eigenvalue of $\tilde{A}$ and a projection of $(\underset{\sim}{u}_L, \underset{\sim}{u}_R)$ on to the eigenvectors of $\tilde{A}$ will be solely on to eigenvectors which correspond to S. Roe [15].

We now calculate coefficients α_i, such that $\Delta\underset{\sim}{u} = \underset{\sim}{u}_L - \underset{\sim}{u}_R$ can be projected onto the eigenvectors $\underset{\sim}{e}$ of A. For $\underset{\sim}{u}_L, \underset{\sim}{u}_R$ close to some average state $\underset{\sim}{u}$, we can write

$$\Delta\underset{\sim}{u} = \sum_1^4 \alpha_i \underset{\sim}{e}_i \tag{5.7}$$

A routine calculation yields

$$\alpha_1 = \frac{1}{2a^2}(\Delta p - a\rho\Delta u) \tag{5.8a}$$

$$\alpha_2 = \rho\Delta v \tag{5.8b}$$

$$\alpha_3 = \Delta\rho - \frac{\Delta p}{a^2} \tag{5.8c}$$

$$\alpha_4 = \frac{1}{2a^2}(\Delta p + a\rho\ \Delta u) \qquad (5.8d)$$

to $O(\Delta^2)$ where $\Delta(\cdot) = (\cdot)_L - (\cdot)_R$.

It can be easily checked that

$$\Delta \underset{\sim}{F} = \sum_1^4 \lambda_i\ \alpha_i\ \underset{\sim}{e}_i \ . \qquad (5.9)$$

As in Roe and Pike [4], we consider the problem of finding average states of the variables such that equations (5.7) and (5.9) hold for the eigenvectors and eigenvalues of the approximate Jacobian $\tilde{A}$ where $\underset{\sim}{u}_L$, $\underset{\sim}{u}_R$ are not necessarily close.

From

$$\Delta \underset{\sim}{u} = \sum_1^4 \tilde{\alpha}_i\ \tilde{\underset{\sim}{e}}_i \qquad (5.10)$$

where $\tilde{\underset{\sim}{e}}_i$ are the eigenvectors of the approximate Jacobian $\tilde{A}$, we find that

$$\tilde{\alpha}_1 = \frac{1}{2\tilde{a}^2}(\Delta p - \tilde{a}\tilde{\rho}\Delta u) \qquad (5.11a)$$

$$\tilde{\alpha}_2 = \tilde{\rho}\Delta v \qquad (5.11b)$$

$$\tilde{\alpha}_3 = \Delta\rho - \frac{\Delta p}{\tilde{a}^2} \qquad (5.11c)$$

$$\tilde{\alpha}_4 = \frac{1}{2\tilde{a}^2}(\Delta p + \tilde{a}\tilde{\rho}\Delta u) \qquad (5.11d)$$

to $O(\Delta^2)$. Note that the tilda above certain variables indicates that they are averaged variables.

We have not yet specified how we are to average the variables. To calculate the averages, we stipulate that equation (5.9) must hold for the average states

i.e.

$$\underset{\sim}{\Delta F} = \sum_{i=1}^{4} \tilde{\lambda}_i \, \tilde{\alpha}_i \, \tilde{e}_i \tag{5.12}$$

Manipulation of these conditions leads to the averages

$$\tilde{u} = \frac{\rho_R^{\frac{1}{2}} u_R + \rho_L^{\frac{1}{2}} u_L}{\rho_R^{\frac{1}{2}} + \rho_L^{\frac{1}{2}}} \tag{5.13a}$$

$$\tilde{\rho} = \rho_L^{\frac{1}{2}} \rho_R^{\frac{1}{2}} \tag{5.13b}$$

$$\tilde{v} = \frac{\rho_R^{\frac{1}{2}} v_R + \rho_L^{\frac{1}{2}} v_L}{\rho_R^{\frac{1}{2}} + \rho_L^{\frac{1}{2}}} \tag{5.13c}$$

$$\tilde{H} = \frac{\rho_R^{\frac{1}{2}} H_R + \rho_L^{\frac{1}{2}} H_L}{\rho_R^{\frac{1}{2}} + \rho_L^{\frac{1}{2}}} \tag{5.13d}$$

and $\tilde{a} = (\gamma-1)\,(\tilde{H}-\tfrac{1}{2}\tilde{q}^2)$ (5.13e)

$$\tilde{q}^2 = \tilde{u}^2 + \tilde{v}^2 \tag{5.13f}$$

and

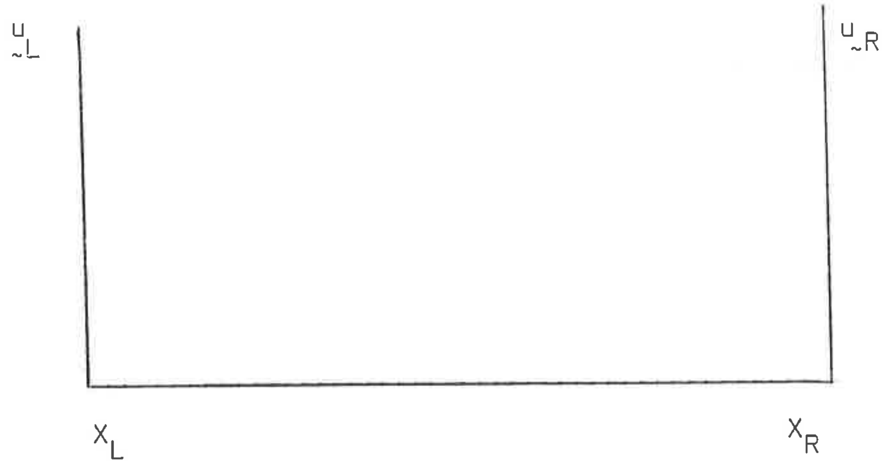
$$\tilde{\lambda}_i = \lambda_i(\tilde{\underset{\sim}{u}}) \tag{5.13g}$$

$$\tilde{\underset{\sim}{e}}_i = \underset{\sim}{e}_i(\tilde{\underset{\sim}{u}}) \tag{5.13h}$$

In this way it is possible to construct a decomposition of $\tilde{A}$ such that properties 1) → 4) above hold.

Similar results hold for the Jacobian $B(\underset{\sim}{u})$, by symmetry.

We can now apply a two-dimensional approximate Riemann solver for the Euler equations using the technique of operator splitting. Using the above results for the decomposition, together with the one-dimensional scalar algorithm given in [4], we perform a sequence of one-dimensional calculations along computational grid lines in the x and y directions in turn. The algorithm along the line y = constant is fully described as follows:- Suppose at time level n we have given states at the right and left hand ends of a computational cell, given by $\underset{\sim}{u}_R$, $\underset{\sim}{u}_L$. Then for each j we update $\underset{\sim}{u}$ to time level $n+1$ in an upwind manner as follows:-



$$\underset{\sim}{u}_R^{n+1} = \underset{\sim}{u}_R^{n} - \frac{\Delta t}{\Delta x} \tilde{\lambda}_j \tilde{\alpha}_j \tilde{\underset{\sim}{e}}_j \qquad \tilde{\lambda}_j > 0$$

$$\underset{\sim}{u}_L^{n+1} = \underset{\sim}{u}_L^{n} - \frac{\Delta t}{\Delta x} \tilde{\lambda}_j \tilde{\alpha}_j \tilde{\underset{\sim}{e}}_j \qquad \tilde{\lambda}_j < 0$$

$$j = 1,2,3,4$$

where $\Delta x = x_R - x_L$ and Δt is the time step.

For the solution operator L_x , $\tilde{\lambda}_i$, $\tilde{\alpha}_i$ and $\tilde{\underset{\sim}{e}}_i$ are given by

$$\tilde{\lambda}_1 = \tilde{u} - \tilde{a}$$

$$\tilde{\lambda}_2 = \tilde{u}$$

$$\tilde{\lambda}_3 = \tilde{u}$$

$$\tilde{\lambda}_4 = \tilde{u} + \tilde{a}$$

$$\tilde{\alpha}_1 = \frac{1}{2\tilde{a}^2} (\Delta p - \tilde{a}\tilde{\rho}\Delta u)$$

$$\tilde{\alpha}_2 = \tilde{\rho} \ \Delta v$$

$$\tilde{\alpha}_3 = \Delta\rho - \frac{\Delta p}{\tilde{a}^2}$$

$$\tilde{\alpha}_4 = \frac{1}{2\tilde{a}^2} (\Delta p + \tilde{a}\tilde{\rho}\Delta u)$$

$$\tilde{\underset{\sim}{e}}_1 = [1, \ \tilde{u}-\tilde{a}, \ \tilde{v}, \ \tilde{H}-\tilde{u}\tilde{a}]^T$$

$$\tilde{\underset{\sim}{e}}_2 = [0, \ 0, \ 1, \ \tilde{v}]^T$$

$$\tilde{\underset{\sim}{e}}_3 = [1, \ \tilde{u}, \ \tilde{v}, \ \tfrac{1}{2}\tilde{q}^2]^T$$

$$\tilde{\underset{\sim}{e}}_4 = [1, \ \tilde{u}+\tilde{a}, \ \tilde{v}, \ \tilde{H}+\tilde{u}\tilde{a}]^T$$

and for the solution operator L_y , $\tilde{\lambda}_i$, $\tilde{\underset{\sim}{e}}_i$, $\tilde{\alpha}_i$

are given by

$$\tilde{\lambda}_1 = \tilde{v} - \tilde{a}$$

$$\tilde{\lambda}_2 = \tilde{v}$$

$$\tilde{\lambda}_3 = \tilde{v}$$

$$\tilde{\lambda}_4 = \tilde{v} + \tilde{a}$$

$$\tilde{\alpha}_1 = \frac{1}{2\tilde{a}^2} (\Delta p - \tilde{a}\tilde{\rho}\Delta v)$$

$$\tilde{\alpha}_2 = \Delta\rho - \frac{\Delta p}{\tilde{a}^2}$$

$$\tilde{\alpha}_3 = \tilde{\rho}\Delta u$$

$$\tilde{\alpha}_4 = \frac{1}{2\tilde{a}^2}(\Delta p + \tilde{a}\tilde{\rho}\Delta v)$$

$$\tilde{\underset{\sim}{e}}_1 = [1,\ \tilde{u},\ \tilde{v}-\tilde{a},\ \tilde{H}-\tilde{v}\tilde{a}]^T$$

$$\tilde{\underset{\sim}{e}}_2 = [1,\ \tilde{u},\ \tilde{v},\ \tfrac{1}{2}\tilde{q}^2]^T$$

$$\tilde{\underset{\sim}{e}}_3 = [0,\ 1,\ 0,\ \tilde{u}]^T$$

$$\tilde{\underset{\sim}{e}}_4 = [1,\ \tilde{u},\ \tilde{v}+\tilde{a},\ \tilde{H}+\tilde{v}\tilde{a}]^T$$

The above algorithm has been used on the test problems described in section 6 and the results are shown in section 7.

6. Two Test Problems

In this section we describe two standard two-dimensional test problems. The first is an infinite spherically divergent shock for which the exact solution is known, and the second is an extension into two dimensions of the standard shocktube problem of Sod [9] and represents a bursting spherical membrane.

The Infinite Spherically Divergent Shock

This test problem has been considered by Noh [1] and by Glaister [2]. Both authors have considered the problem in one space dimension using cylindrical geometry. Noh treated the problem by introducing artificial viscosity and artificial heat flux and compared his method with the standard non-Neumann-Richtmeyer artificial viscosity method [10], Schulz's tensor Q formulation [11] and Woodward and Collela's P.P.M. [12]. Glaister treated the problem using a spherically symmetric extension of the standard linearised approximate Riemann solver. Both authors have shown that, although this test problem has a very simple solution (see [1] and [3]), difficulties arise in calculating good numerical results due to an instantaneous infinite pressure jump at the origin. However, the methods employed by both authors have proved to be efficient at following the shock in the solution.

The problem begins with flow of a gas radially and into the origin such that the speed of the gas at any point is Mach 1.0. Initially density and pressure are everywhere uniform and constant and the pressure is zero. The gas is reflected at the origin and expands outwards.

We introduce reflective boundaries along $x = 0$ and $y = 0$ (to simulate the radial symmetry of the problem) and maintain the exact solution on the outflow boundaries.

The equations of motion governing the flow are the two dimensional Euler equations, namely

$$\underset{\sim}{u}_t + \underset{\sim}{F}(\underset{\sim}{u})_x + \underset{\sim}{G}(\underset{\sim}{u})_y = 0 \tag{6.1}$$

where $\underset{\sim}{F}$, $\underset{\sim}{G}$ and $\underset{\sim}{u}$ have been defined in section 2 and the gas constant, γ, is taken to be 5/3. The initial conditions are

$$\rho(x,y,0) = 1.0 \tag{6.2a}$$

$$u(x,y,0) = -x/_R \tag{6.2b}$$

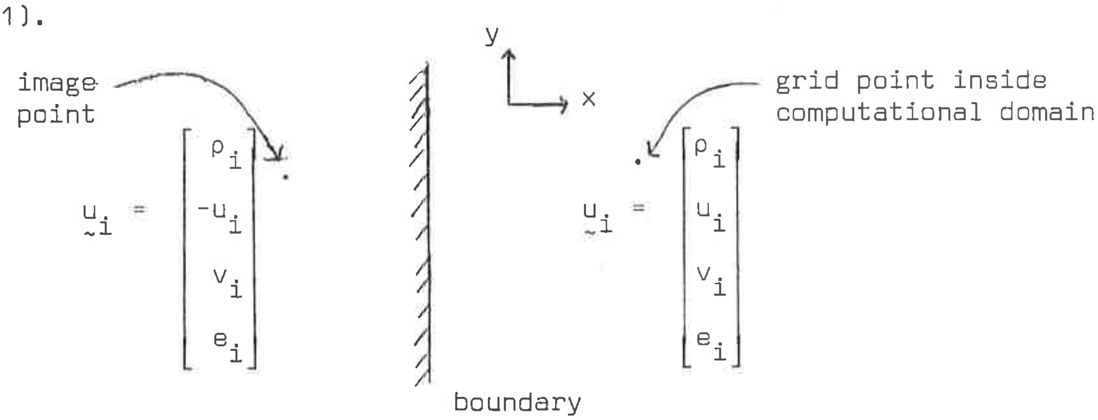
$$v(x,y,0) = -y/_R \tag{6.2c}$$

$$e(x,y,0) = 0.5 \tag{6.2d}$$

where R is the radius from the origin, $R = \sqrt{x^2+y^2}$.

As can be seen from (6.2d), (2.5) and (2.6), the initial pressure is zero.

To implement the boundary conditions we consider flow to be reflected conservatively, i.e. flow tangential to the boundary is unaltered whilst flow normal to the boundary is reflected using a method of images, (see figure 1).



$$\underset{\sim}{u}_i = \begin{bmatrix} \rho_i \\ -u_i \\ v_i \\ e_i \end{bmatrix} \qquad \underset{\sim}{u}_i = \begin{bmatrix} \rho_i \\ u_i \\ v_i \\ e_i \end{bmatrix}$$

Figure 1

This guarantees that there is no flow out of the region along Ox or Oy and that on the boundary the normal component of flow is zero.

The exact solution for this problem is one involving an infinite divergent shock radiating from the origin with uniform velocity $s = 1/_3$. Post-shock values are

$$\rho^+ = 16.0 \tag{6.3a}$$

$$u^+ = 0.0 \tag{6.3b}$$

$$v^+ = 0.0 \tag{6.3c}$$

$$e^+ = 8.0 \tag{6.3d}$$

$$p^+ = 5.3\dot{3} \tag{6.3e}$$

and pre-shock values:

$$\rho^- = 1.0 + T/_R \tag{6.4a}$$

$$u^- = -x/_R \tag{6.4b}$$

$$v^- = -y/_R \tag{6.4c}$$

$$e^- = 0.5 \tag{6.4d}$$

$$p^- = 0.0 \tag{6.4e}$$

where T is the time. See figure 2.

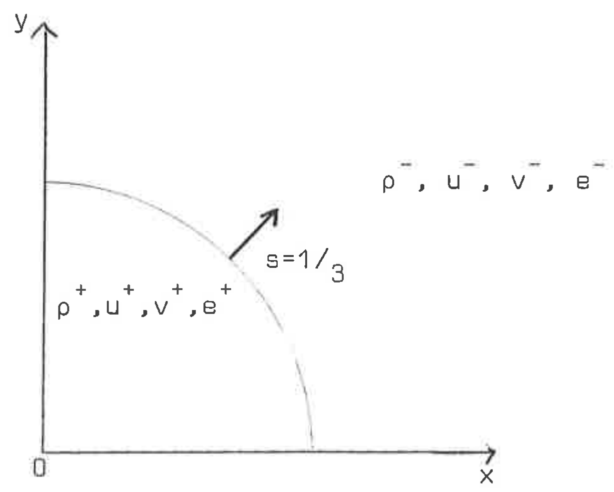


figure 2

As can be seen, the jump in pressure, p^+/p^- , is infinite and instantaneous. It is due to this that many difficulties have been encountered in obtaining good numerical solutions.

The Bursting Membrane

This test problem has been considered by Glaister [2] using a one dimensional cylindrically symmetric extension of Roe's scheme. In many respects, this problem is much simpler to obtain good numerical results for, since the shock is present in the initial data and there are no infinite jumps in any of the variables.

Here we are dealing with a two-dimensional gas lying, initially, at rest in a region. Initially the region is divided by a circular membrane of radius R . There are finite jumps across the membrane in both density and pressure.

At time $t = 0$, the membrane is removed (or burst). Shockwaves form and move towards the origin.

Again we implement reflection boundary conditions along Ox and Oy .

At some later time a rapidly moving shockwave, which has reflected from the origin, interacts with a slower moving one that has not already reached the origin.

The equations of motion governing the flow are the two-dimensional Euler equations (6.1). The gas constant, γ , is taken to be 1.4. The initial conditions are

$$\left.\begin{array}{lll} \rho_i & = & 1.0 \\ u_i & = & 0.0 \\ v_i & = & 0.0 \\ p_i & = & 0.4 \end{array}\right\} \quad r < R$$

$$\left.\begin{array}{lll} \rho_o & = & 4.0 \\ u_o & = & 0.0 \\ v_o & = & 0.0 \\ p_o & = & 1.6 \end{array}\right\} \quad r > R$$

where $r = \sqrt{x^2+y^2}$

We assume transparent boundary conditions on the outflow boundaries, that is, on these boundaries the variables are kept at their initial states of ρ_o, u_o, v_o, p_o, and the solution is not allowed to run to a time such that the shockwaves would reach these boundaries. The initial conditions are shown in figure 3.

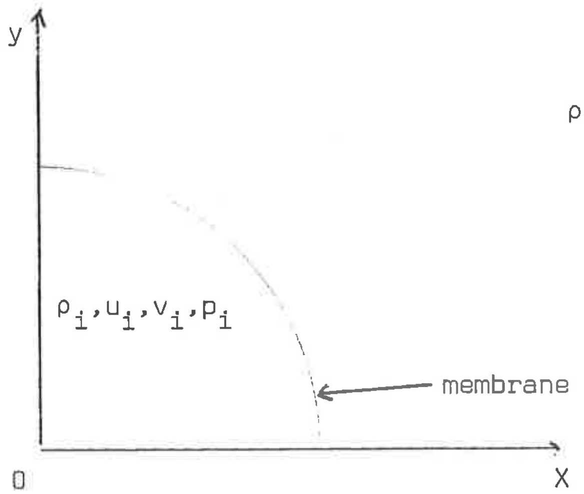


figure 3

7. Results of Numerical Tests

The algorithm described in earlier sections using operator splitting and Roe decomposition was used to solve both the aforementioned test problems. Although the algorithm is only first order accurate, surprisingly good results have been obtained for both problems.

The computational domain was taken to be $[0,1] \times [0,1]$ and for all solutions presented here $\Delta x = \Delta y$.

Results for the infinite spherically diverging shock

During the computation of the wave speeds $\tilde{\alpha}_i$ in the approximate Riemann solver we rely on the fact that the sound speed, $\tilde{a}$, is non-zero. However, when we consider the case when the pressure is zero then we have that

$$p = 0$$

$$\Rightarrow \quad p = (\gamma-1)\rho i = 0$$

$$\Rightarrow \quad i = 0 \qquad (\text{since } \gamma \neq 1 \text{ and } \rho \neq 0)$$

and

$$e = \rho i + \tfrac{1}{2}\rho q^2$$

$$= \tfrac{1}{2}\rho q^2$$

thus

$$H = \frac{e+p}{\rho} = \tfrac{1}{2}q^2$$

and hence

$$a^2 = (\gamma-1)\,(H-\tfrac{1}{2}q^2)$$

$$= 0 \ .$$

This is equivalent to saying that the sonic waves in the decomposition, $\tilde{\underset{\sim}{e}}_1$ and $\tilde{\underset{\sim}{e}}_4$, do not travel. Thus we can set $\tilde{\alpha}_1$, $\tilde{\alpha}_4$ equal to zero when the pressure p is equal to zero and also set

$$\tilde{\alpha}_2 = \tilde{\rho}\Delta v$$

$$\tilde{\alpha}_3 = \Delta\rho \quad .$$

Contour projections are given for density with various values of Δx , Δy and Δt . Every second figure is a plot of the variable against distance taken along the line $x = y$ from the origin.

Figures 1 and 2 show the output for density at time 0.6 with

$$\Delta x = \Delta y = 0.02$$

$$\Delta t = 0.005$$

and has a maximum CFL number of 0.15147.

Figures 3 and 4 show the output for density at time 0.6 and

$$\Delta x = \Delta y = 0.01$$

$$\Delta t = 0.0025$$

and has a maximum CFL number of 0.15191.

Figures 5 through to 12 show the output for density every 0.15 seconds with

$$\Delta x = \Delta y = 0.005$$

$$\Delta t = 0.00125$$

and has a final output time of 0.6. Here the maximum CFL number is 0.20259.

Figures 13 to 32 show output for all the conserved variables at output times 0.6 and 1.2 with

$$\Delta x = \Delta y = 0.01$$

$$\Delta t = 0.003.$$

An attempt to make the problem less severe by starting at a non-zero time resulted in figures 33 to 36. The problem was started at time t = 0.18

and was run to the time t = 0.6.

Results for the Bursting Membrane problem

Figures 1 to 10 show the output for density at every 0.11 seconds until a final time of t = 0.55.

$\Delta x = \Delta y = 0.02$

$\Delta t = 0.005$

and the maximum CFL number is 0.26659.

Figures 11 to 34 show output for density at output times t = 0.1375, 0.275, 0.35, 0.4125, 0.45, 0.55.

Figures 11 to 22 are for

$\Delta x = \Delta y = 0.01$

$\Delta t = 0.00125$

and have a maximum CFL number of 0.14161.

Figures 22 to 34 are for

$\Delta x = \Delta y = 0.005$

$\Delta t = 0.00125$

and have a maximum CFL number of 0.33002.

Discussion of Results

As can be seen from the first set of results, the severity of the problem was reflected by poor numerical results, especially at the origin. The density suffered more severely than any of the other variables, supporting its claim to be the most sensitive variable. However, the algorithm managed to track the shock at the correct speed, despite the post shock density being about 25% in error at the origin (Noh records where errors were in excess of 100% at the origin).

The usual features of the underlying first order algorithm are present, namely smoothing of data at the shock interface. Two other interesting features are noted, firstly, spurious contours near the boundaries - which are thought to be due to the sharp velocity gradients along the boundaries, and, secondly, along the pre-shock curve in density, we notice that the numerical solution dips below the exact solution in the region of $R \in [0.8, 1.2]$, this is particularly noticeable in figures 12, and is attributed to the splitting technique employed and may be related to the squaring of curved contours recently encountered by the author in dealing with scalar problems, or reflection of waves from the outflow boundary.

The results for the bursting membrane are comparable to those produced by Glaister and are significantly better than those of the previous test problem. The only noticeable feature is that the solution sags a little more than it ought to. This is noticeable when comparing figure 34 with the results of Glaister, and again it is thought to be due to the operator splitting.

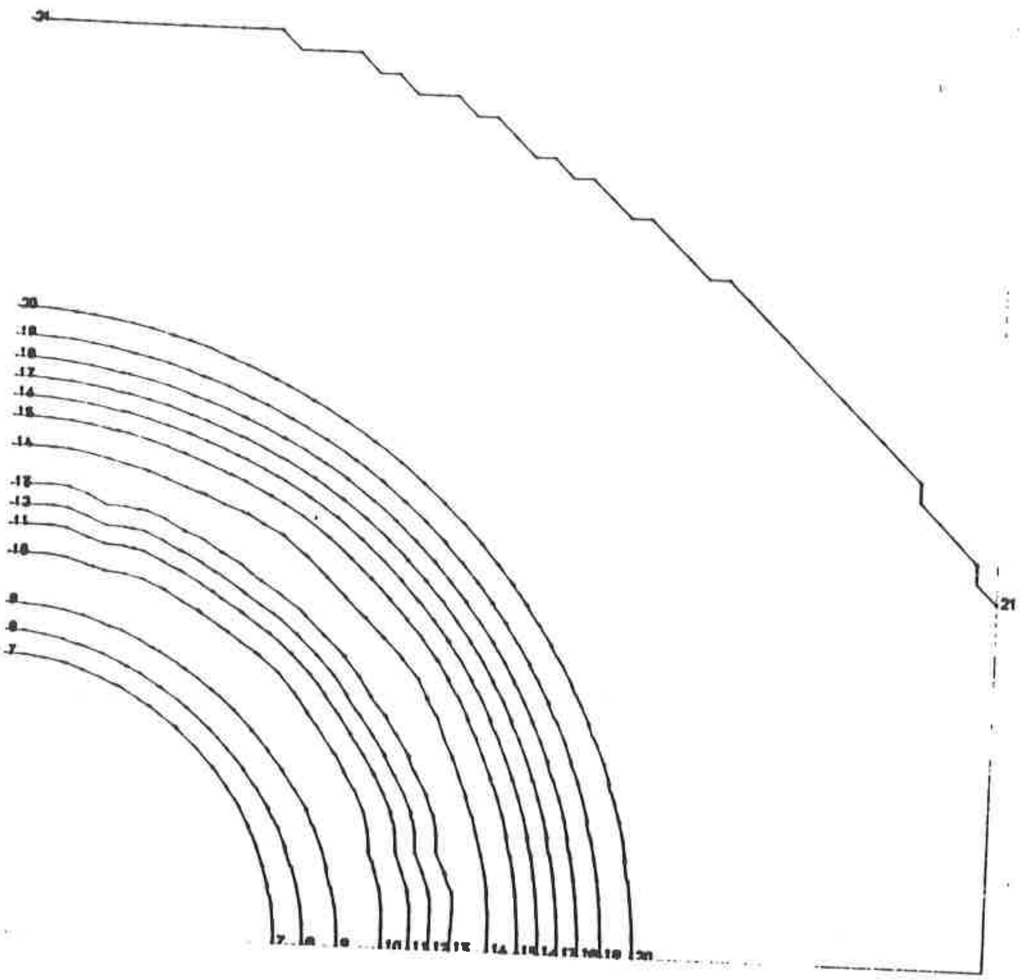
8. Conclusion

We have shown that the standard technique of using a two dimensional Riemann solver incorporating operator splitting applied to the Euler equations can give satisfactory results for the bursting membrane problem and also that results can be achieved via this method for the infinite spherically divergent shock, and that these results are comparable to results attained via other methods.

Future Aims

It has been suggested that the algorithm used be adapted to incorporate limiters (see Sweby [13]) to see if better results could be obtained by using an essentially second order method. However, it is felt by the author that the use of limiters would detract from the accuracy of the solution as it would introduce even more one dimensional effects into the algorithm and this would result in squaring of the contours.

The author is currently researching into genuinely two dimensional algorithms for scalar conservation laws.



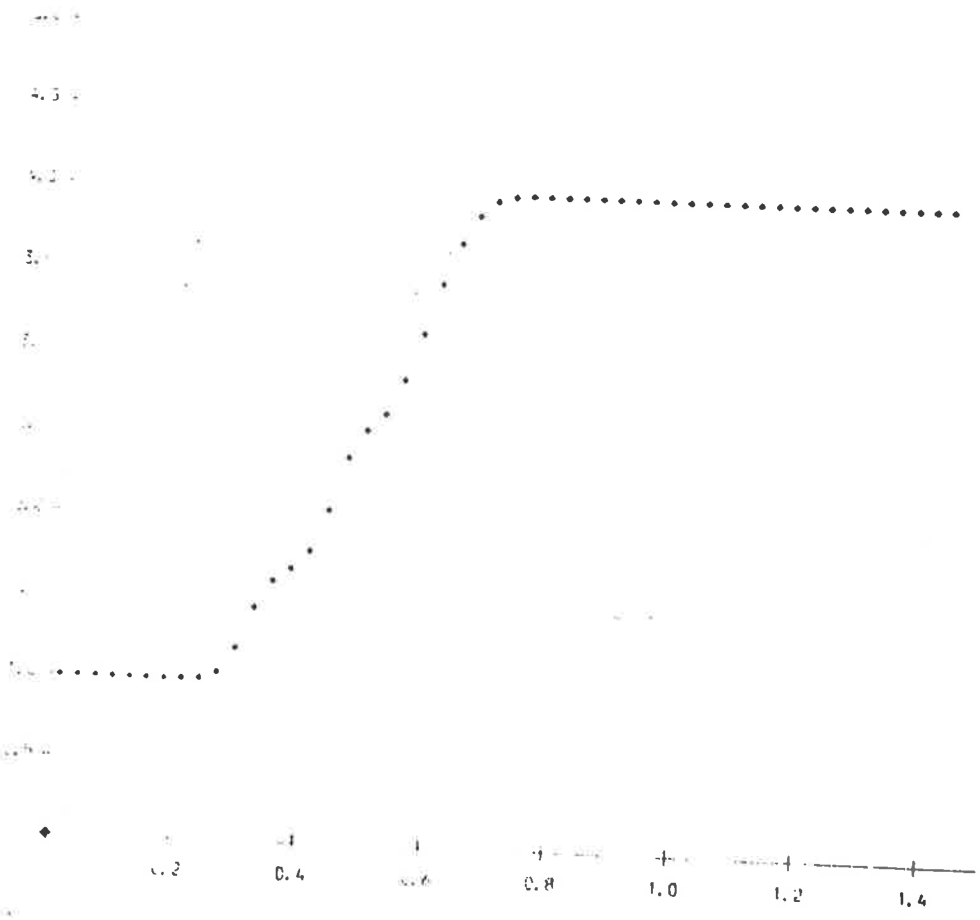
SOLUTION OF SODS PROBLEM
USING [illegible] OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY

TIME = 0.1100
22 TIME STEPS

DT = 0.00500
DX = 0.02000
DY = 0.02000
MAX CFL = 0.23867
MAX HEIGHT = 4.00000
MIN HEIGHT = 0.00000

Figure 1



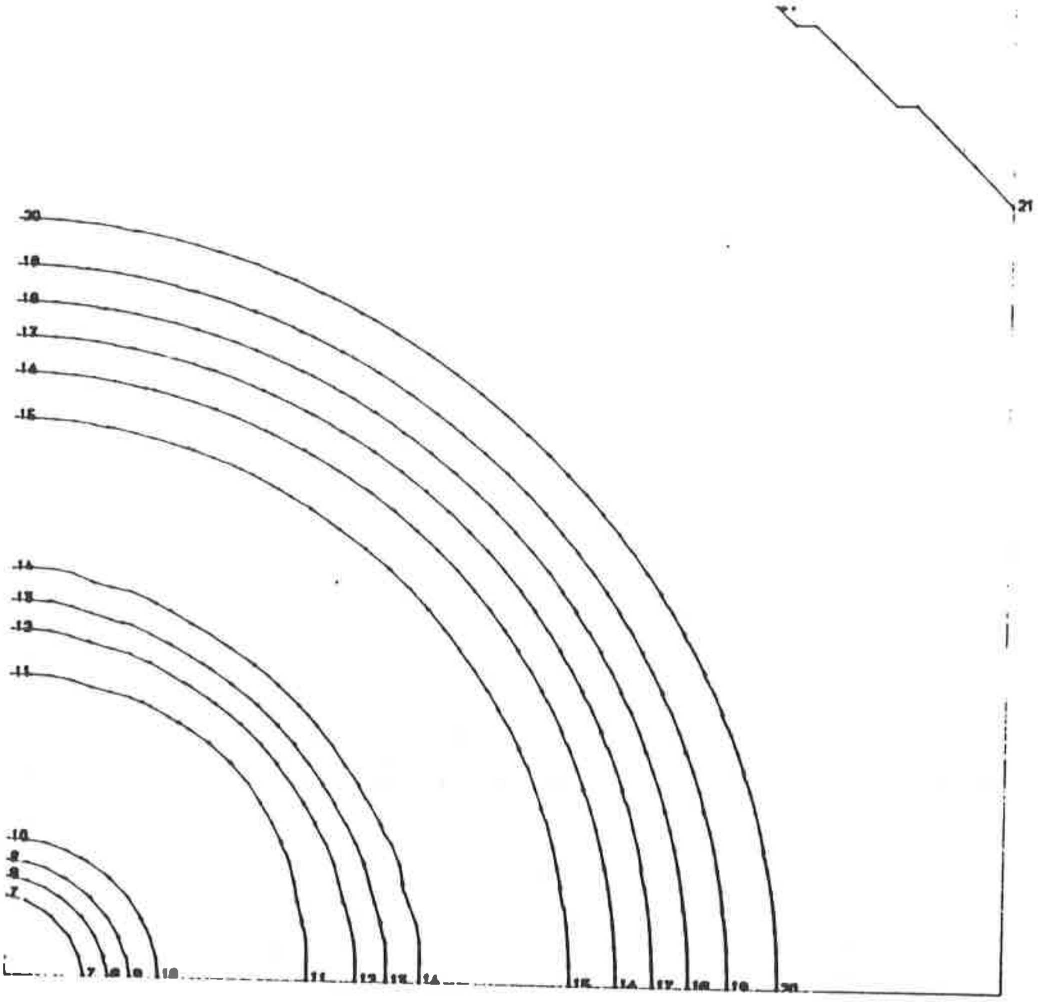
SOLUTION OF SODS PROBLEM
USING [2] OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY

TIME = 0.1100
22 TIME STEPS

DT = 0.00500
DX = 0.02000
DY = 0.02000
MAX CFL = 0.23867
MAX HEIGHT = 4.00000
MIN HEIGHT = 0.00000

Figure 2



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

TIME : 0.2200

44 TIME STEPS

DT = 0.00500

DX = 0.02000

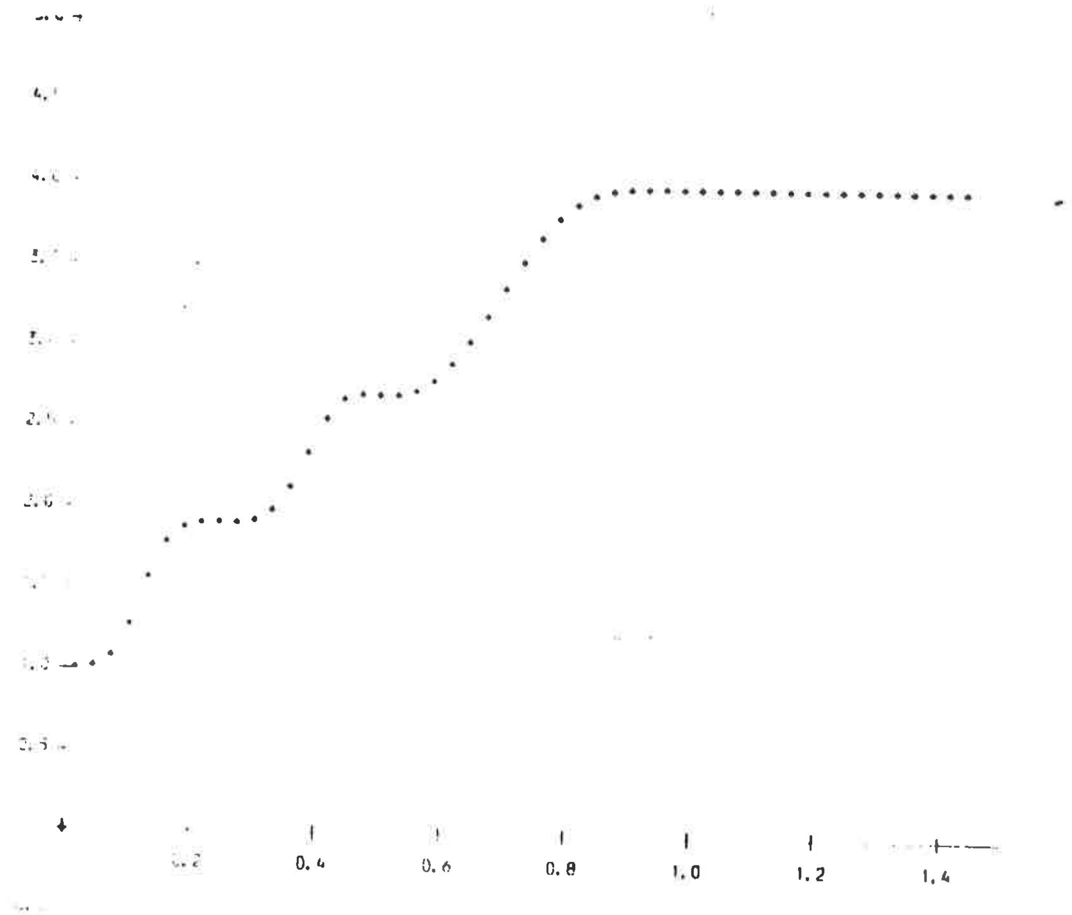
DY = 0.02000

MAX CFL : 0.25663

MAX HEIGHT : 4.00000

MIN HEIGHT : 0.00000

Figure 3



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

TIME : 0.2200

44 TIME STEPS

DT = 0.00500

DX = 0.02000

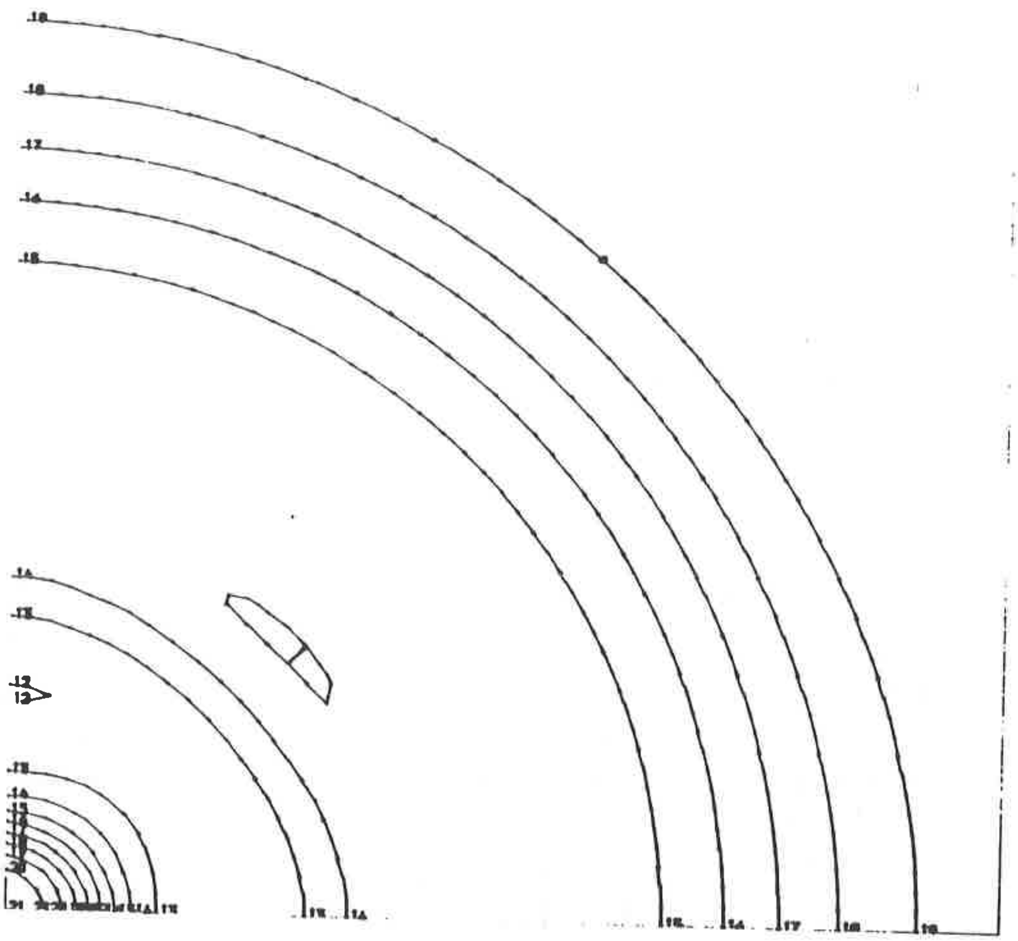
DY = 0.02000

MAX CFL : 0.25663

MAX HEIGHT : 4.00000

MIN HEIGHT : 0.00000

Figure 4



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR .

DENSITY

TIME . 0.3300

66 TIME STEPS

DT = 0.00500

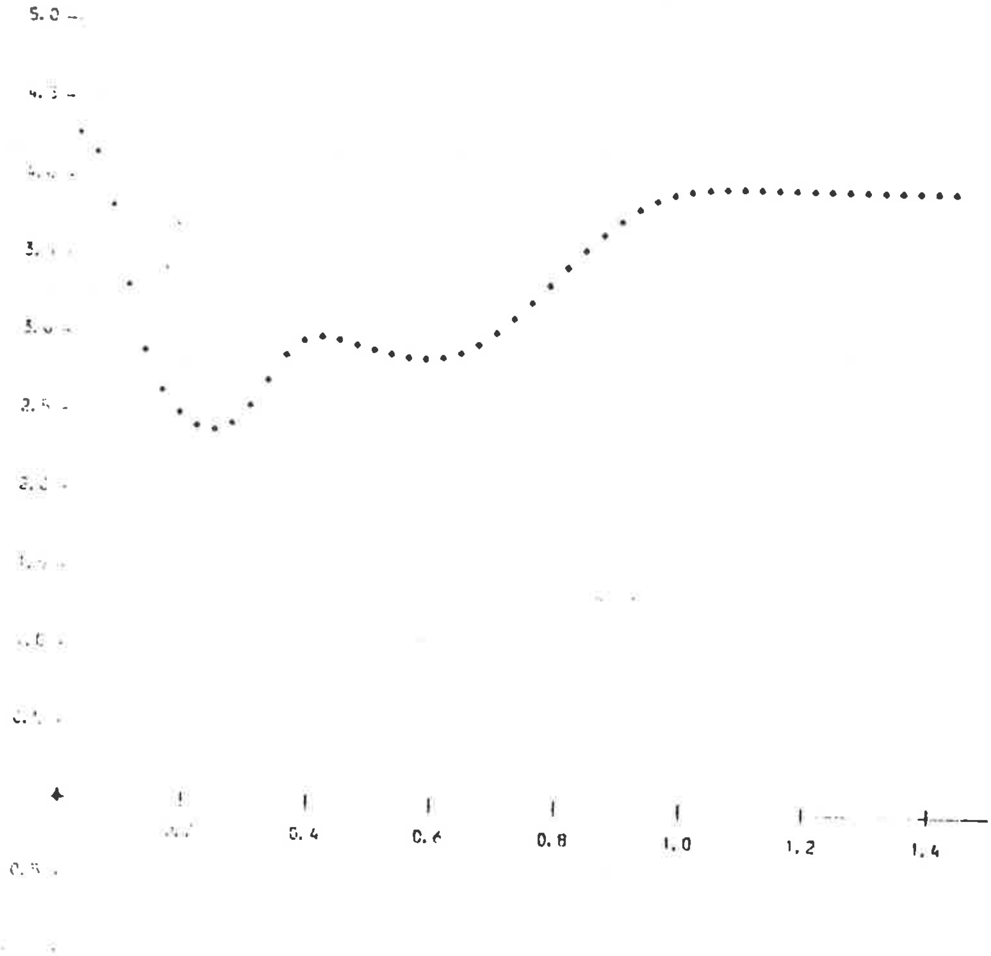
DX = 0.02000

DY = 0.02000

MAX CFL . 0.26659

MAX HEIGHT . 4.27480

MIN HEIGHT . 0.00000



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR .-

DENSITY

TIME . 0.3300

66 TIME STEPS

DT = 0.00500

DX = 0.02000

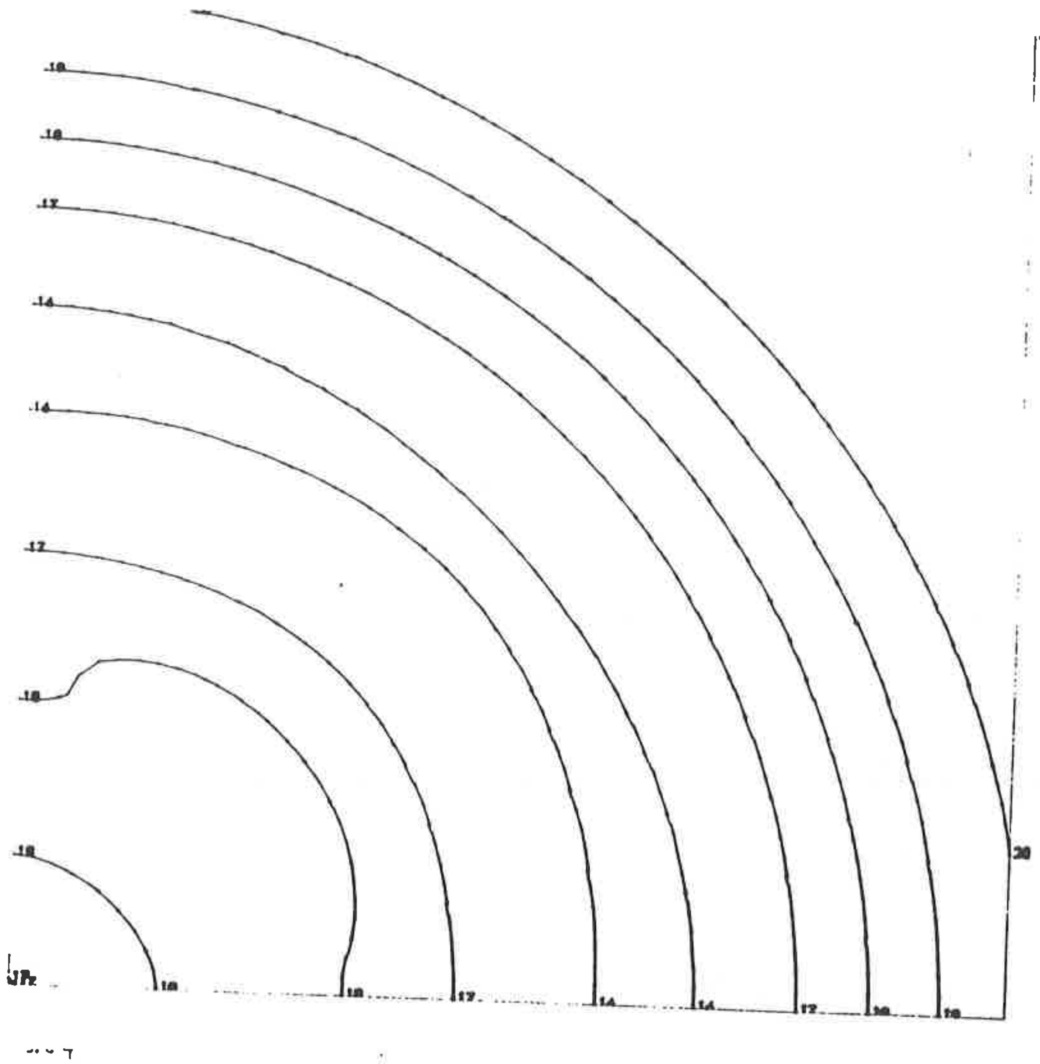
DY = 0.02000

MAX CFL . 0.26659

MAX HEIGHT . 4.27480

MIN HEIGHT . 0.00000

Figure 6



SOLUTION OF SODS PROBLEM

USING [2] OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

TIME : 0.4400

88 TIME STEPS

DT = 0.00500

DX = 0.02000

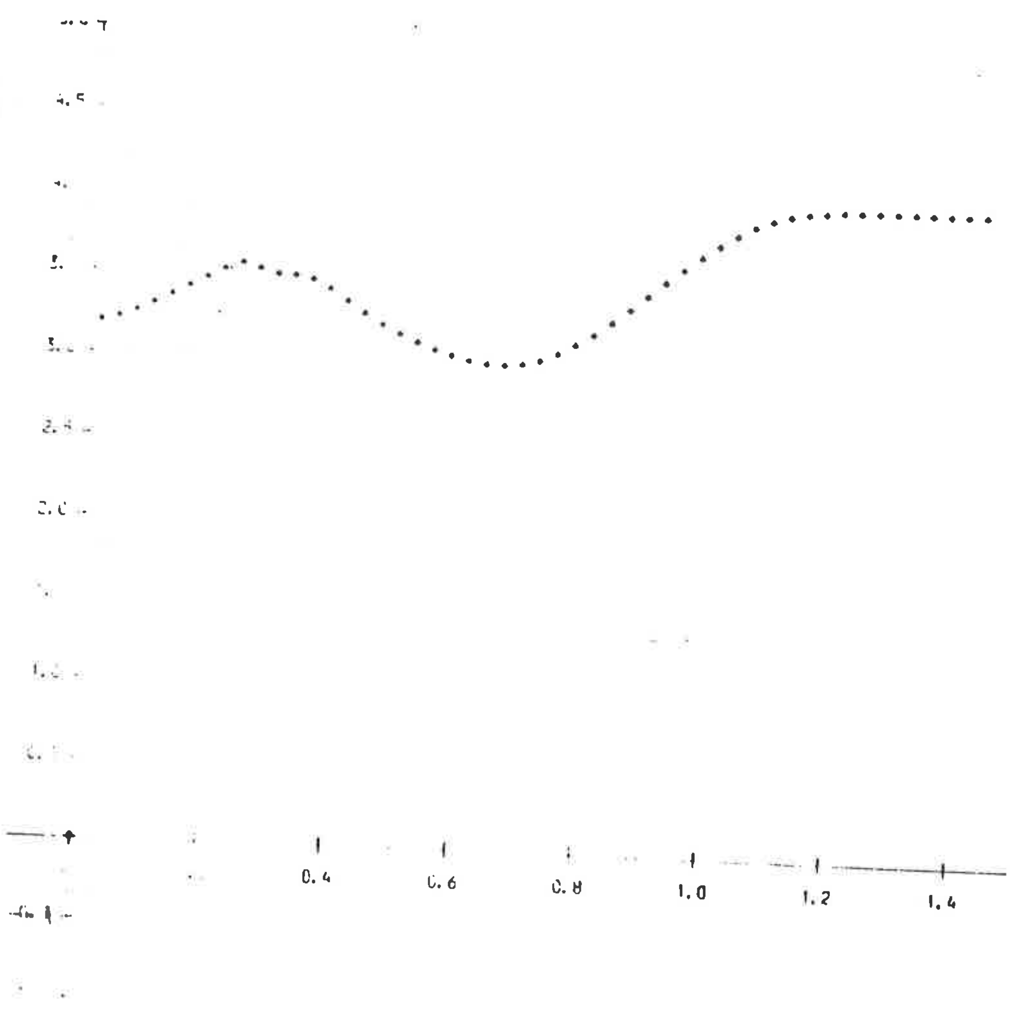
DY = 0.02000

MAX CFL : 0.26659

MAX HEIGHT : 4.00000

MIN HEIGHT : 0.00000

Figure 7



SOLUTION OF SODS PROBLEM

USING [2] OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

TIME : 0.4400

88 TIME STEPS

DT = 0.00500

DX = 0.02000

DY = 0.02000

MAX CFL : 0.26659

MAX HEIGHT : 4.00000

MIN HEIGHT : 0.00000

Figure 8

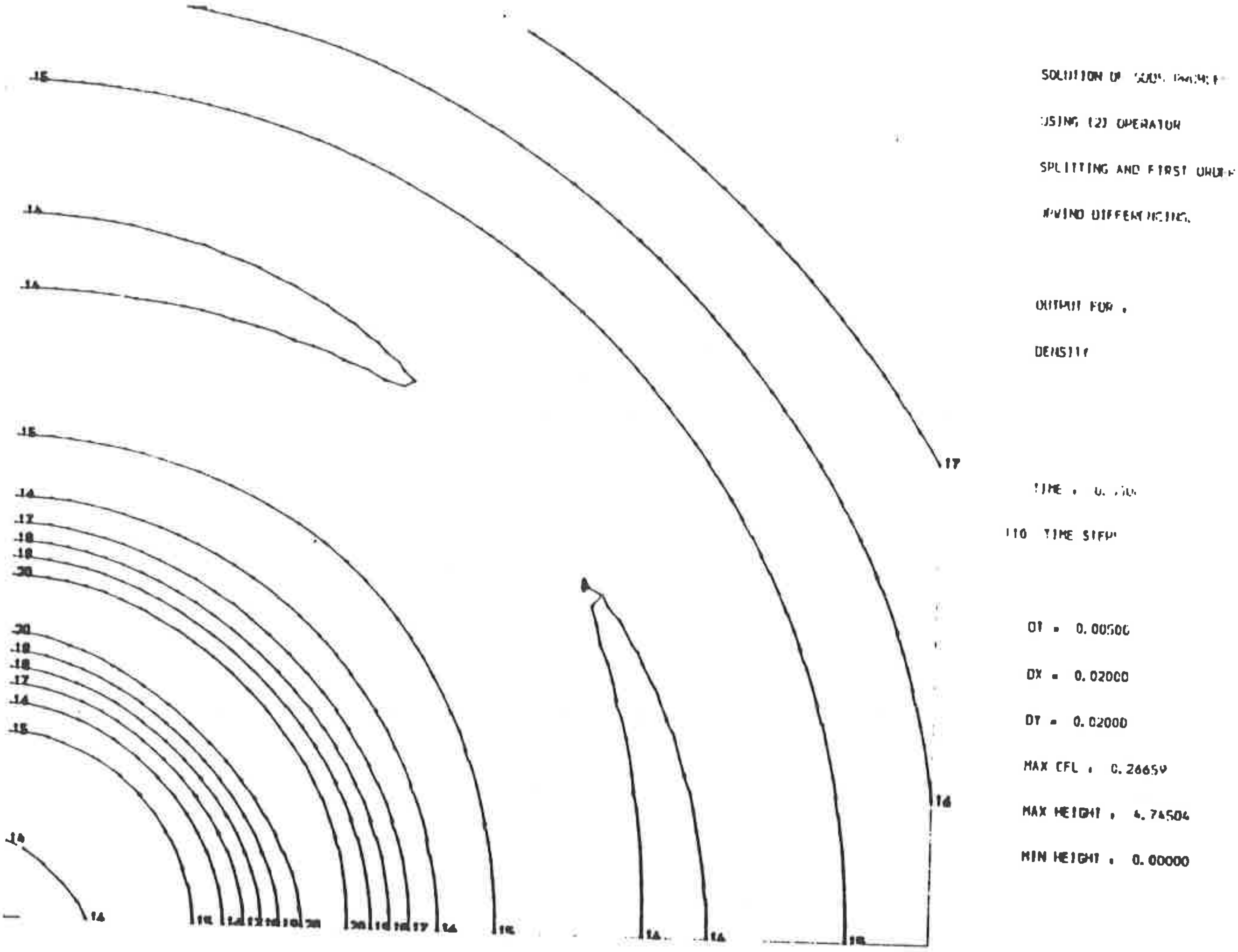


Figure 9

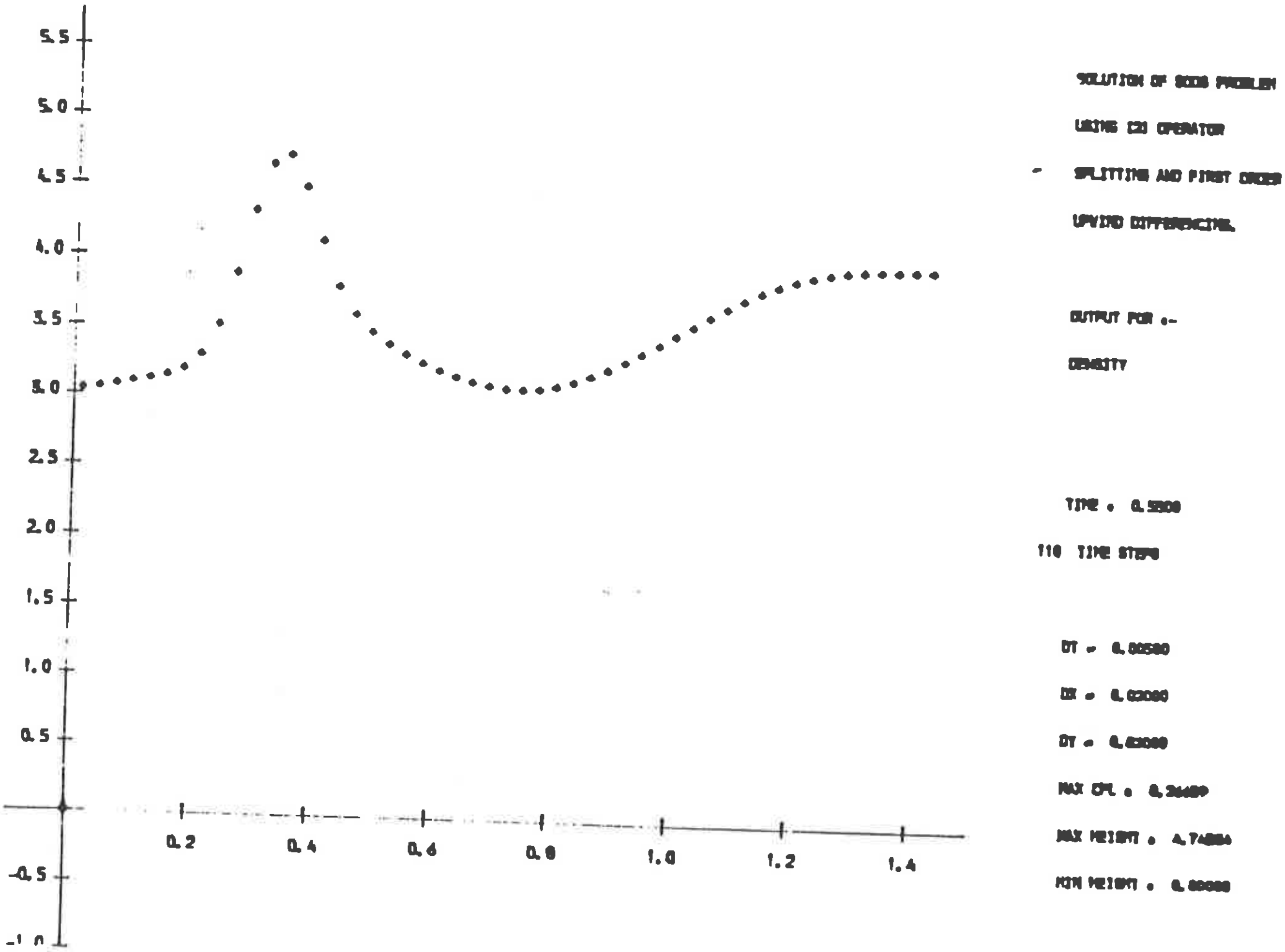
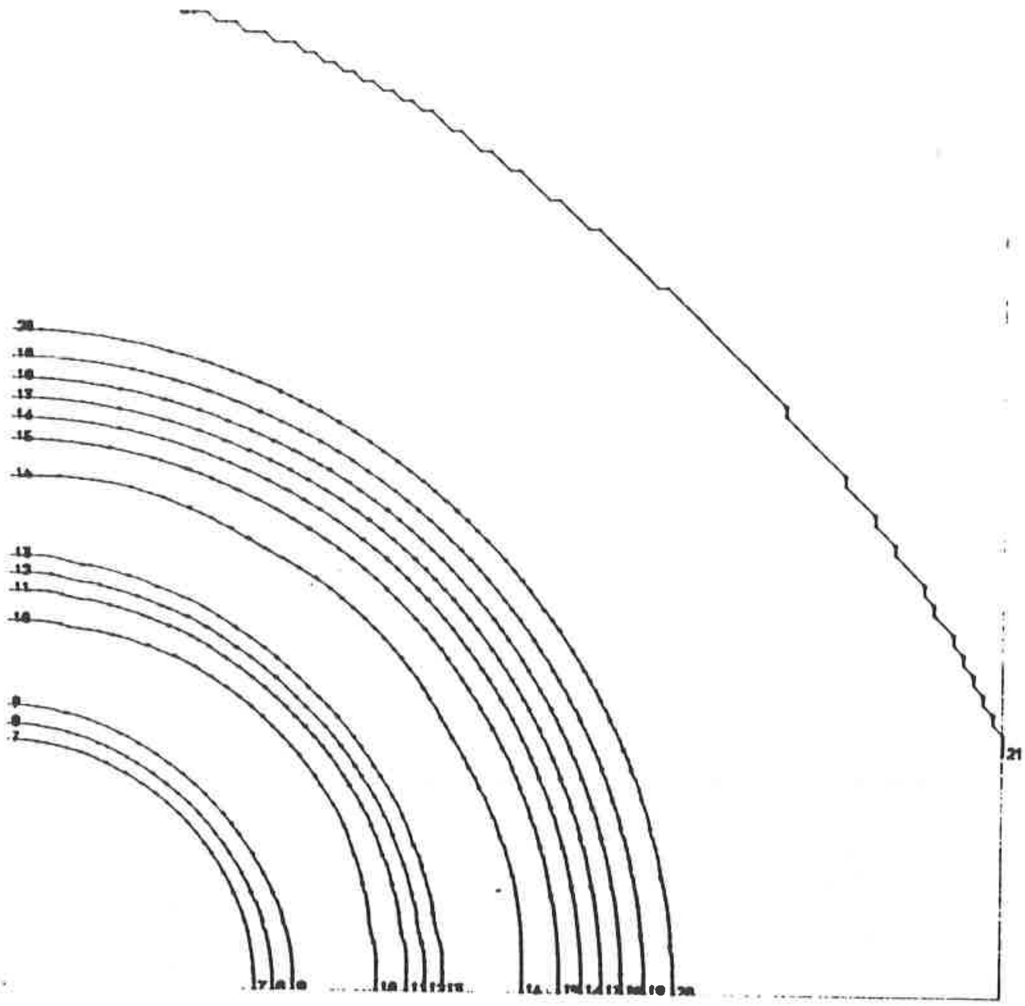


Figure 10



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :

DENSITY

TIME : 0.1375

110 TIME STEPS

DT = 0.00125

DX = 0.01000

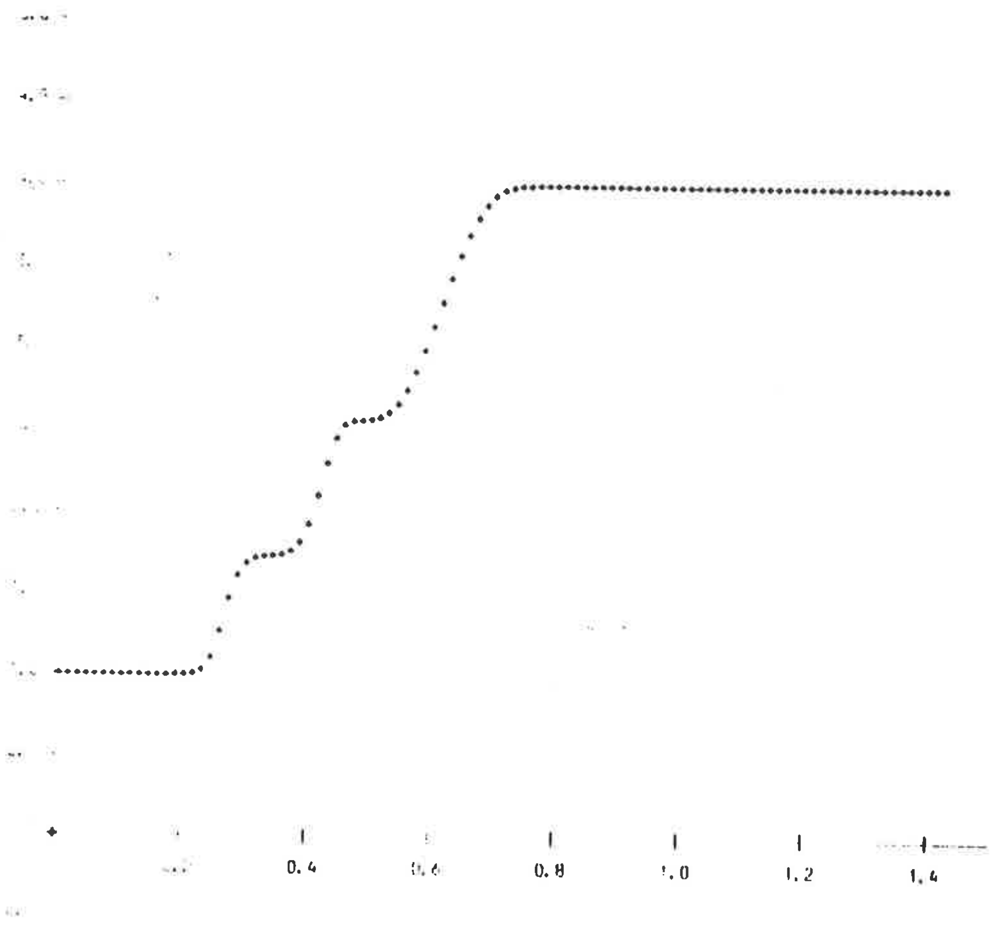
DY = 0.01000

MAX CFL : 0.12361

MAX HEIGHT : 4.00000

MIN HEIGHT : 0.00000

figure 11



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

TIME : 0.1375

110 TIME STEPS

DT = 0.00125

DX = 0.01000

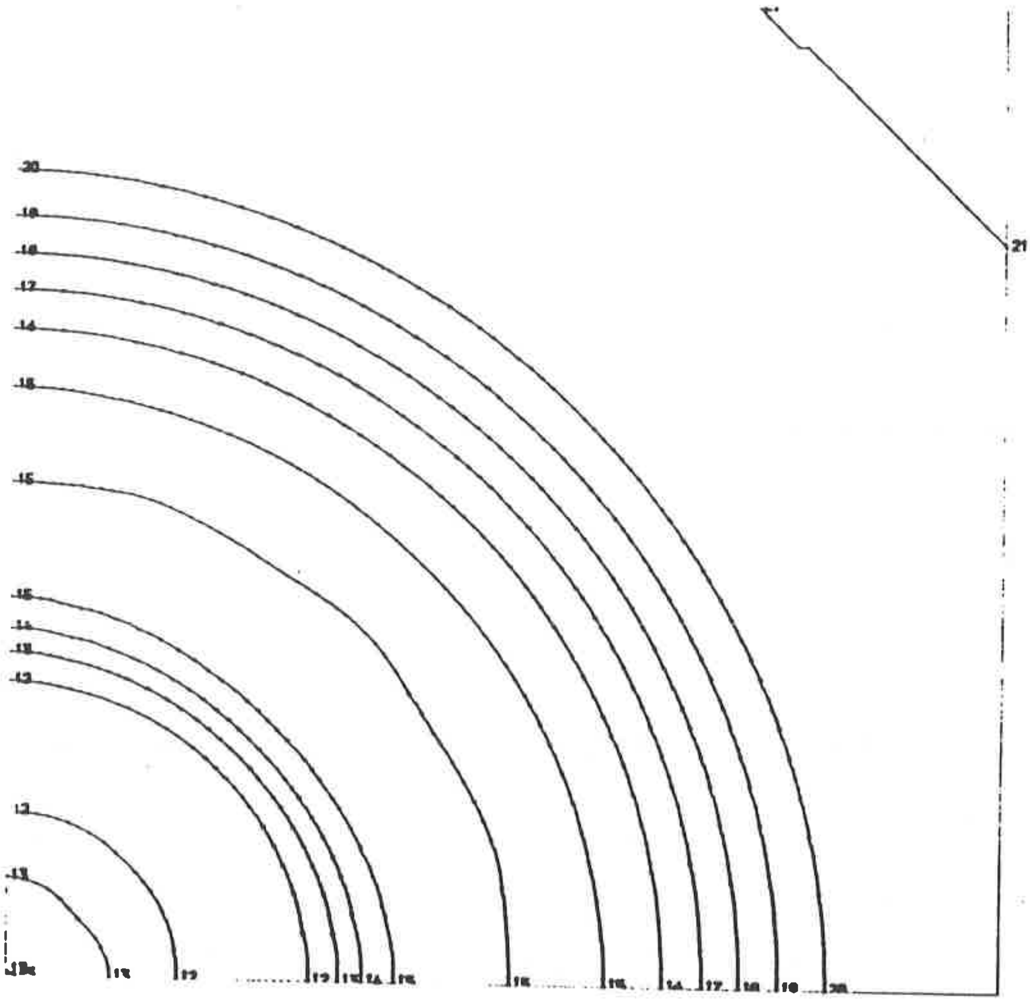
DY = 0.01000

MAX CFL : 0.12361

MAX HEIGHT : 4.00000

MIN HEIGHT : 0.00000

figure 12



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

TIME = 0.2750

220 TIME STEPS

DT = 0.00125

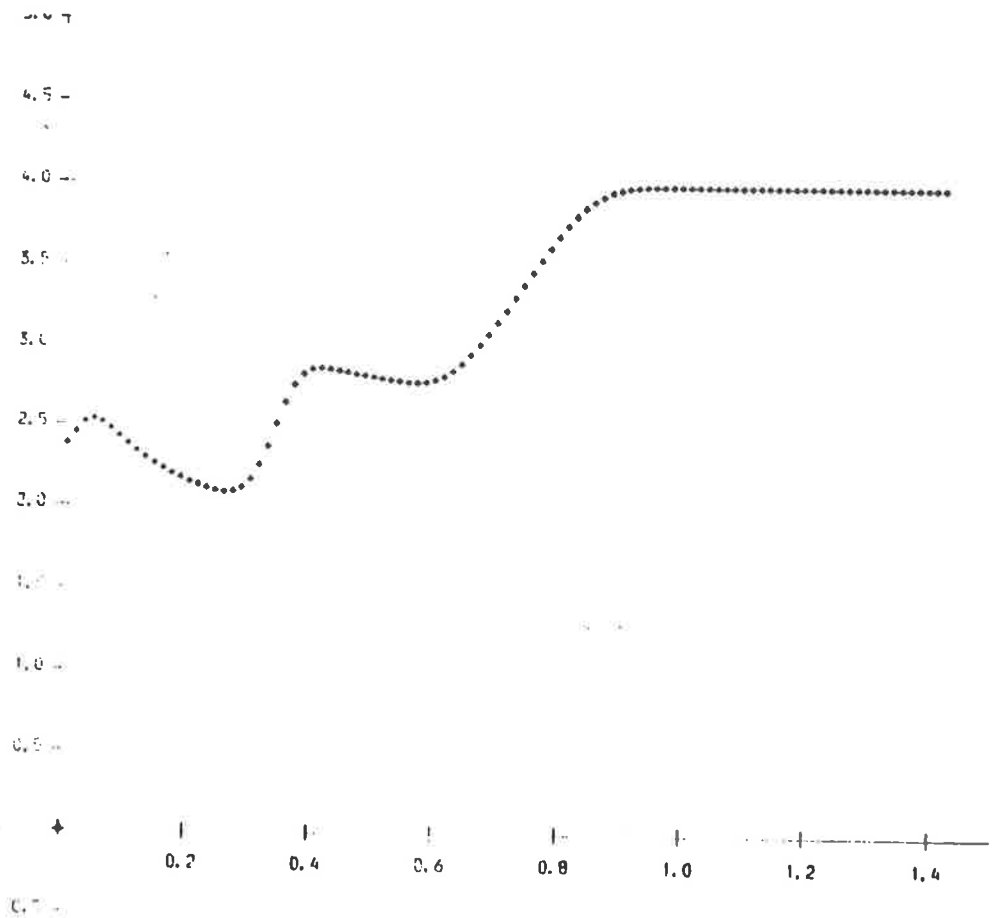
DX = 0.01000

DY = 0.01000

MAX CFL = 0.14139

MAX HEIGHT = 4.00000

MIN HEIGHT = 0.00000



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

TIME = 0.2750

220 TIME STEPS

DT = 0.00125

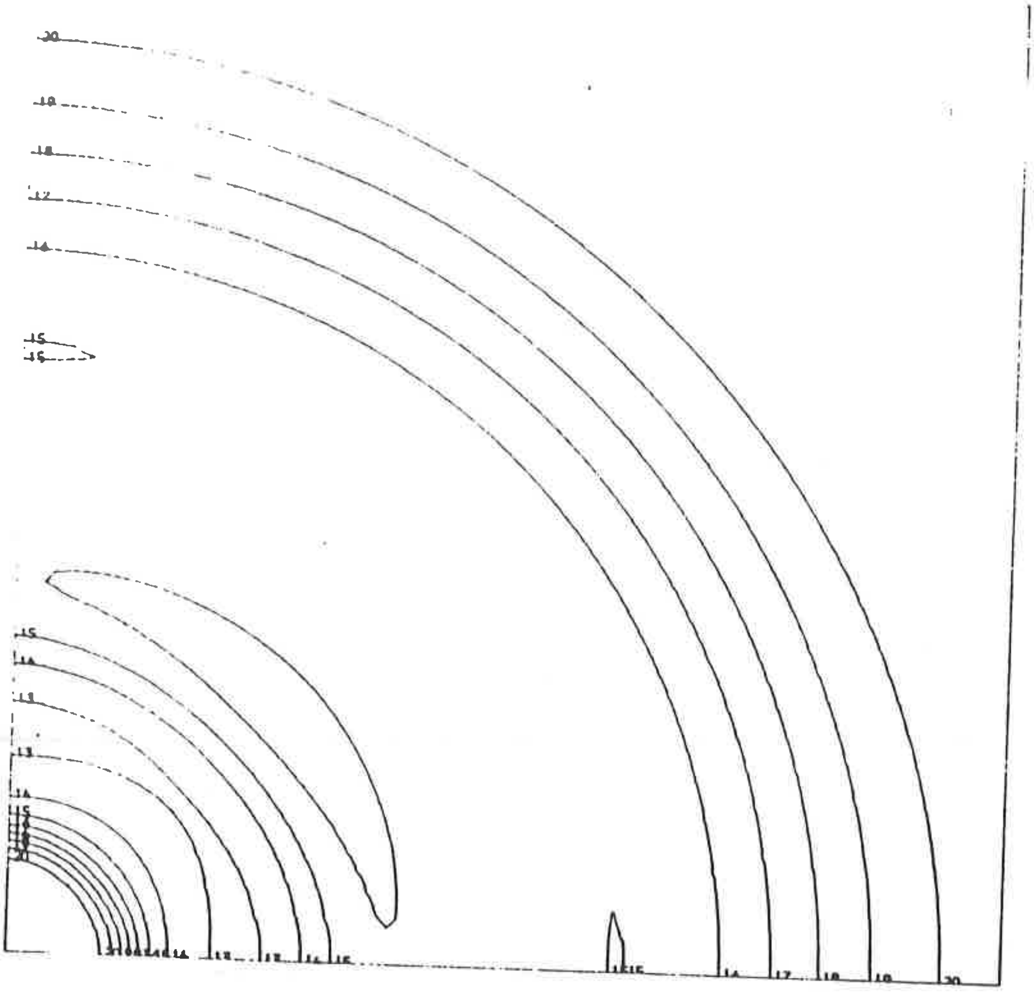
DX = 0.01000

DY = 0.01000

MAX CFL = 0.14139

MAX HEIGHT = 4.00000

MIN HEIGHT = 0.00000



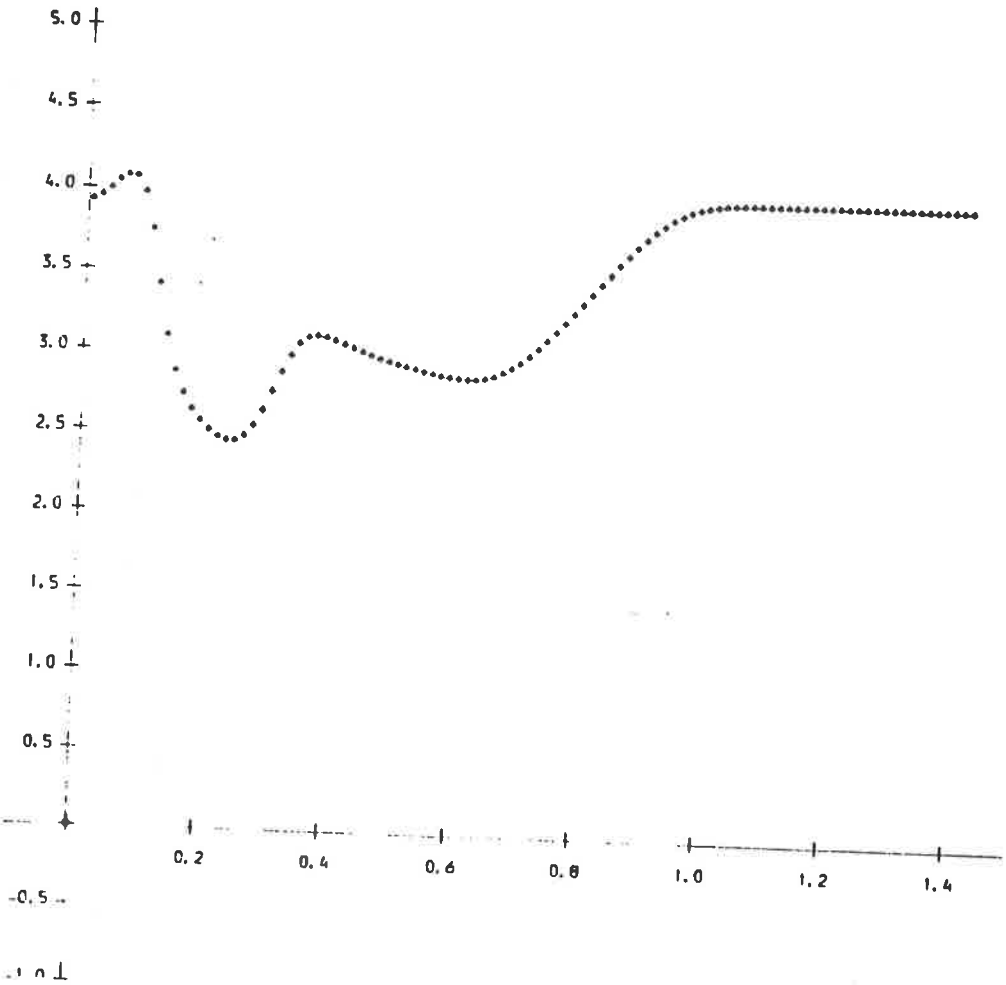
SOLUTION OF SODS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY

TIME : 0.3500
280 TIME STEPS

DT = 0.00125
DX = 0.01000
DY = 0.01000
MAX CFL : 0.14161
MAX HEIGHT : 4.08306
MIN HEIGHT : 0.00000

Figure 15



SOLUTION OF SODS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY

TIME : 0.3500
280 TIME STEPS

DT = 0.00125
DX = 0.01000
DY = 0.01000
MAX CFL : 0.14161
MAX HEIGHT : 4.08306
MIN HEIGHT : 0.00000

Figure 16

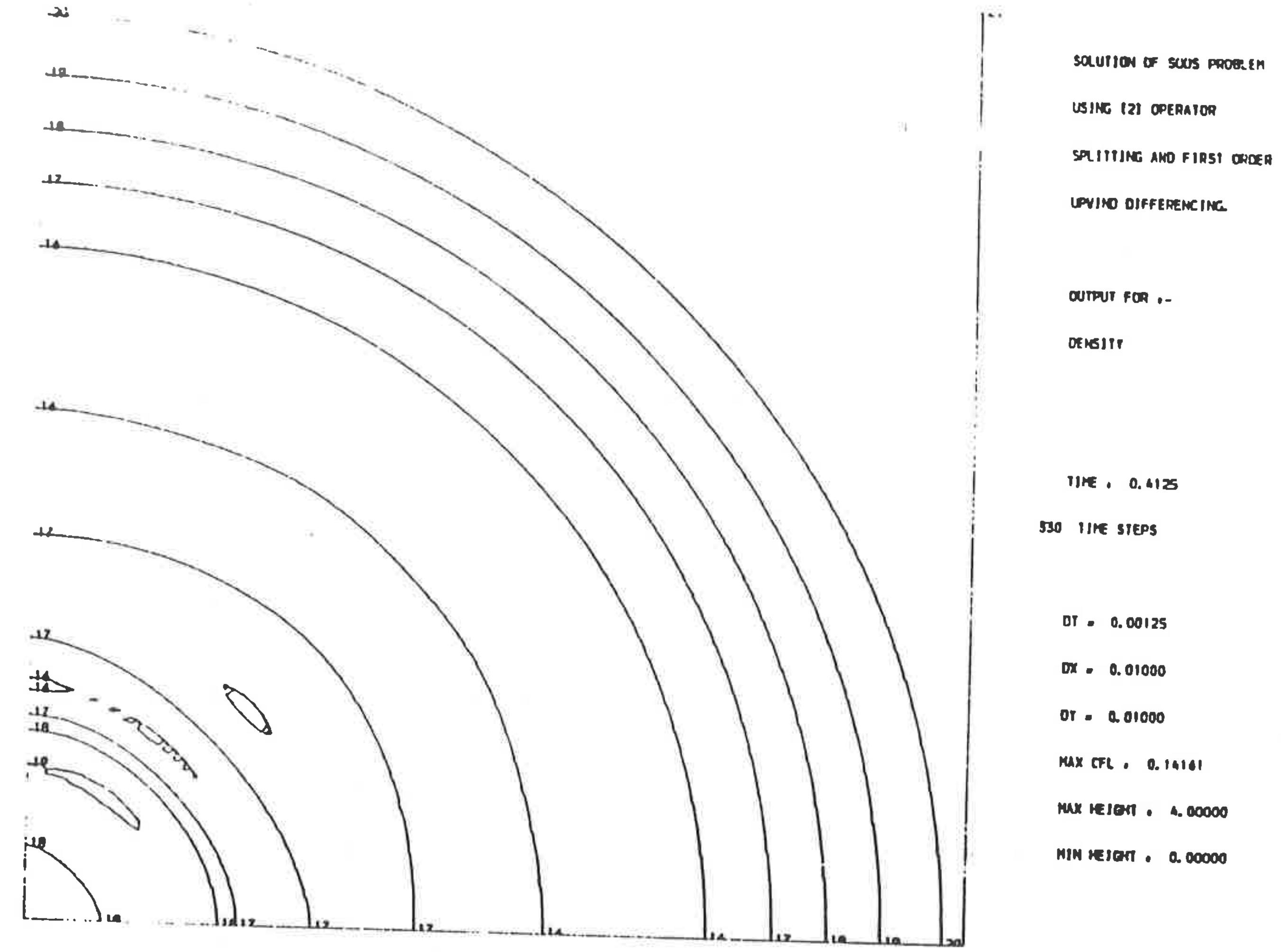


Figure 17

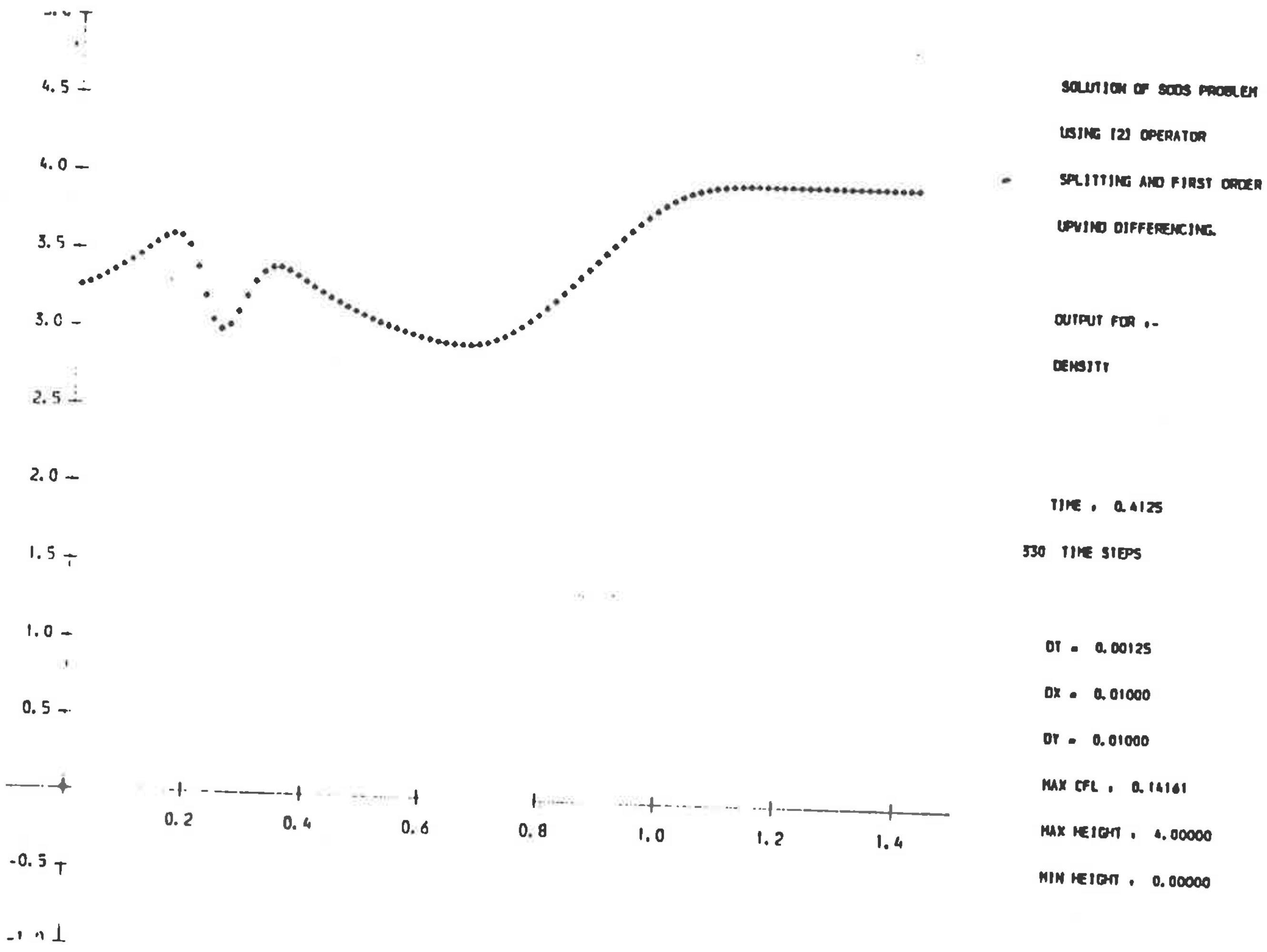
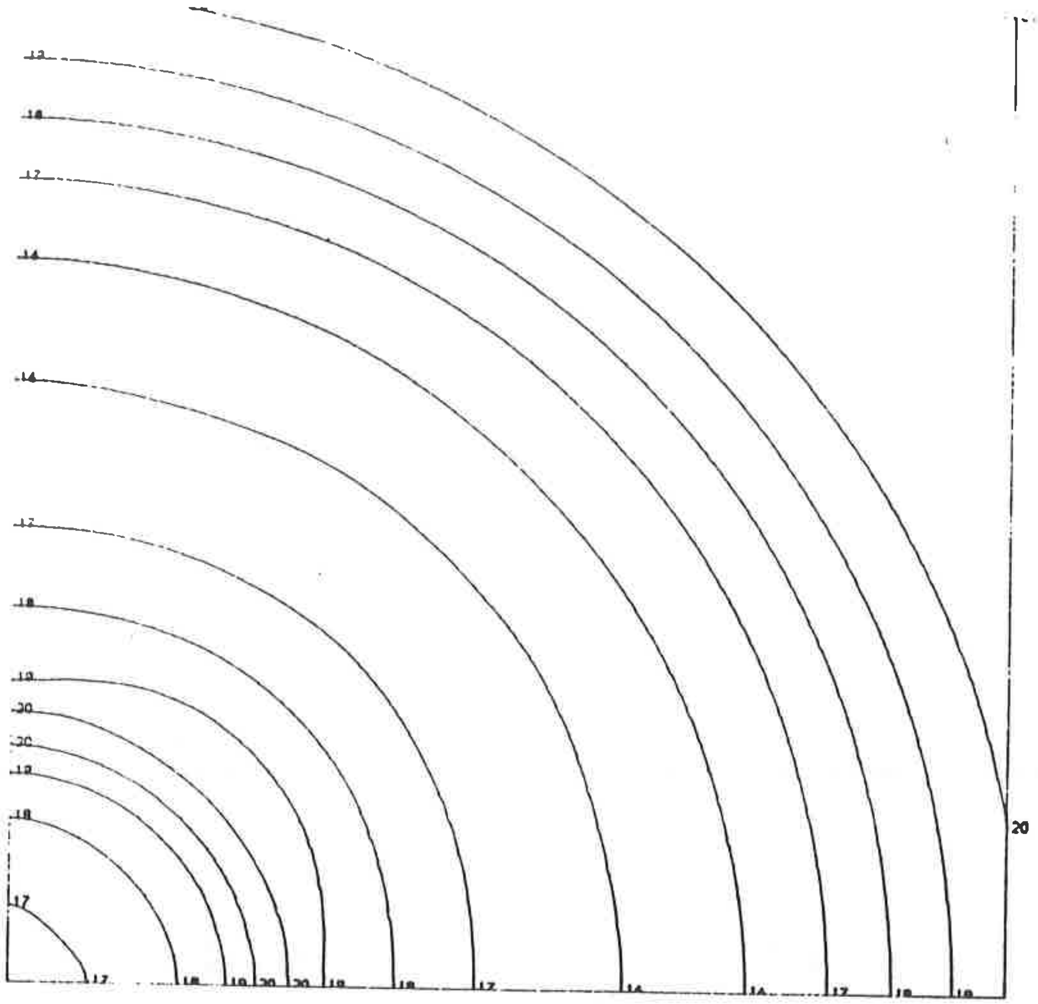


Figure 18



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

TIME : 0.4500

360 TIME STEPS

DT = 0.00125

DX = 0.01000

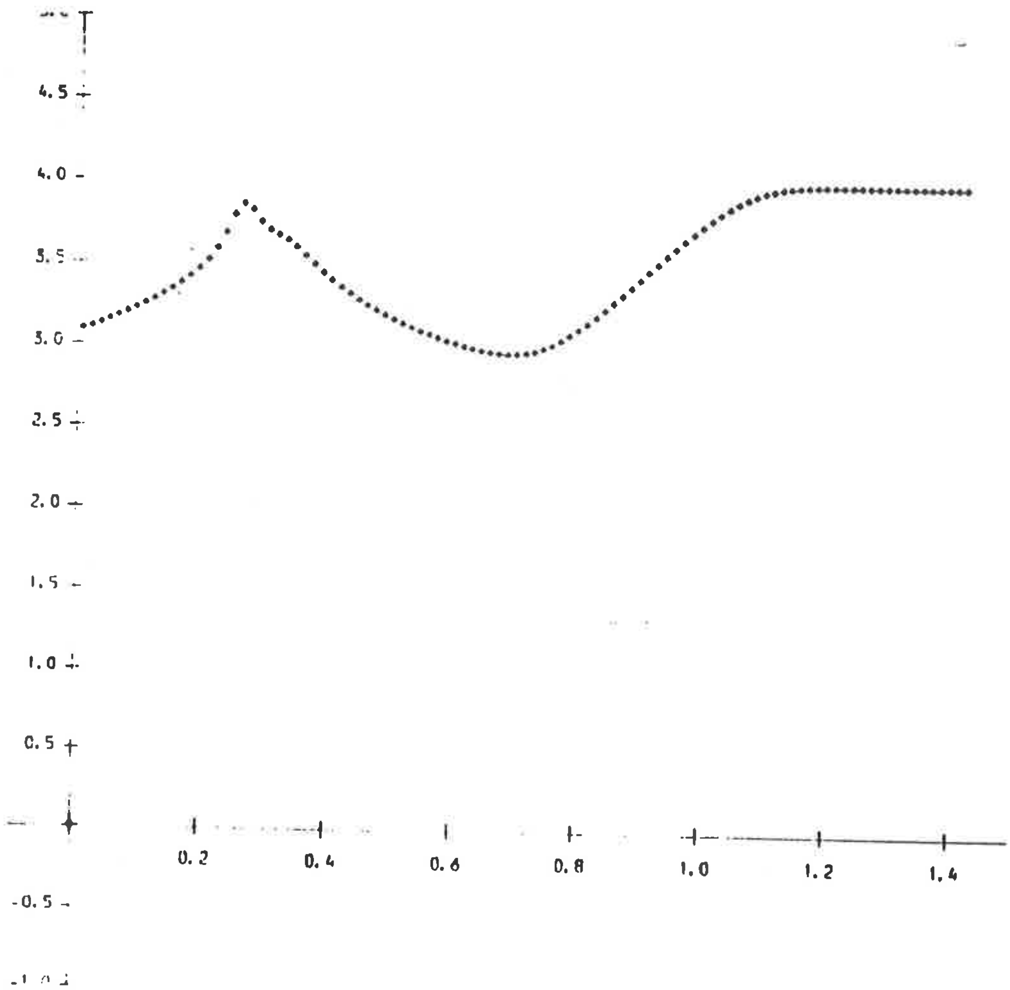
DY = 0.01000

MAX CFL : 0.14161

MAX HEIGHT : 4.00000

MIN HEIGHT : 0.00000

Figure 19



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

TIME : 0.4500

360 TIME STEPS

DT = 0.00125

DX = 0.01000

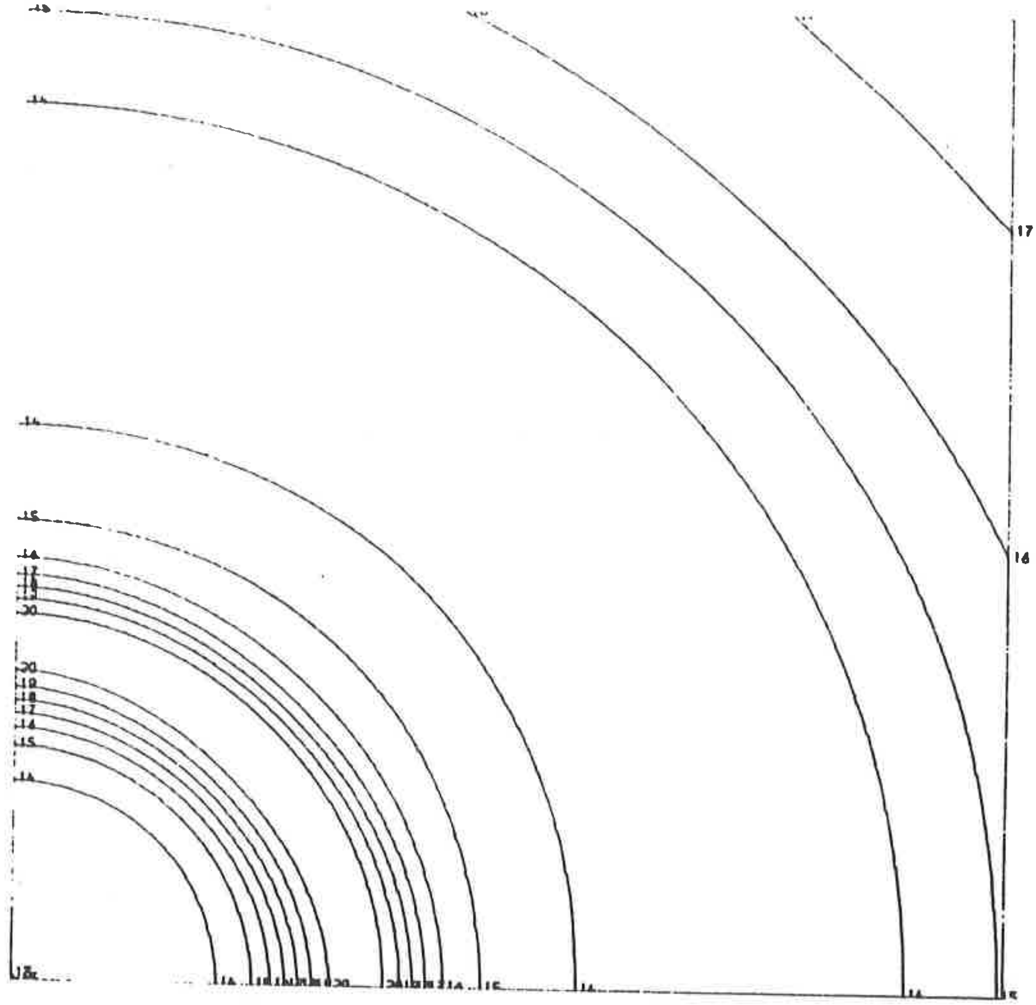
DY = 0.01000

MAX CFL : 0.14161

MAX HEIGHT : 4.00000

MIN HEIGHT : 0.00000

Figure 20



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

TIME : 0.5500

440 TIME STEPS

DT = 0.00125

DX = 0.01000

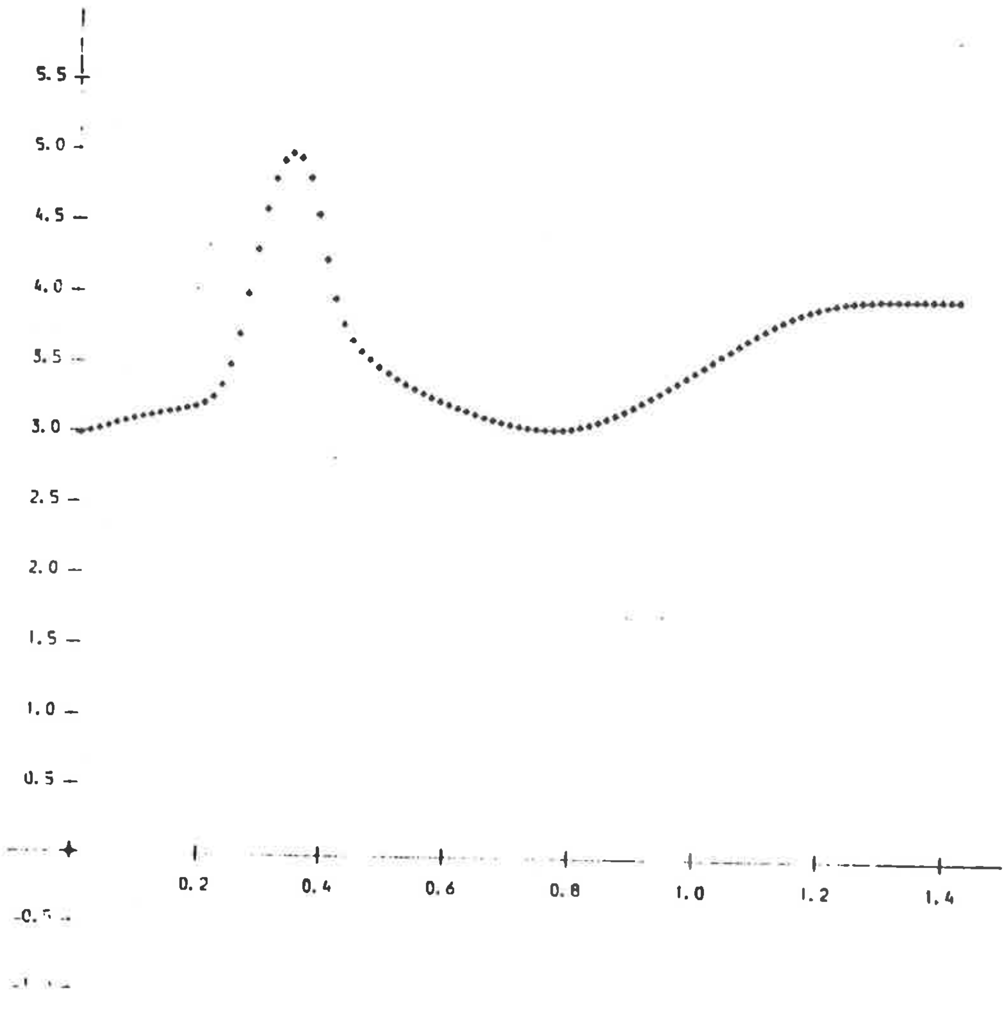
DY = 0.01000

MAX CFL : 0.14161

MAX HEIGHT : 4.98700

MIN HEIGHT : 0.00000

Figure 21



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

TIME : 0.5500

440 TIME STEPS

DT = 0.00125

DX = 0.01000

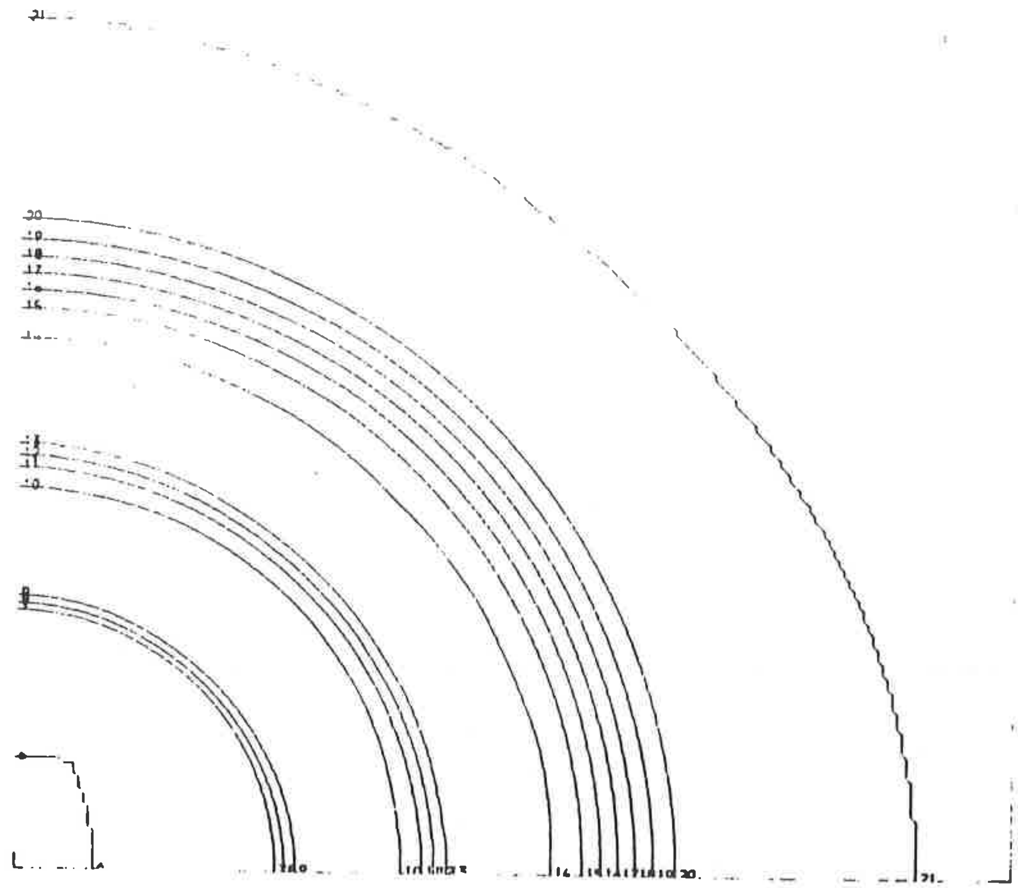
DY = 0.01000

MAX CFL : 0.14161

MAX HEIGHT : 4.98700

MIN HEIGHT : 0.00000

Figure 22



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

TIME : 0.1375

110 TIME STEPS

DT = 0.00125

DX = 0.00500

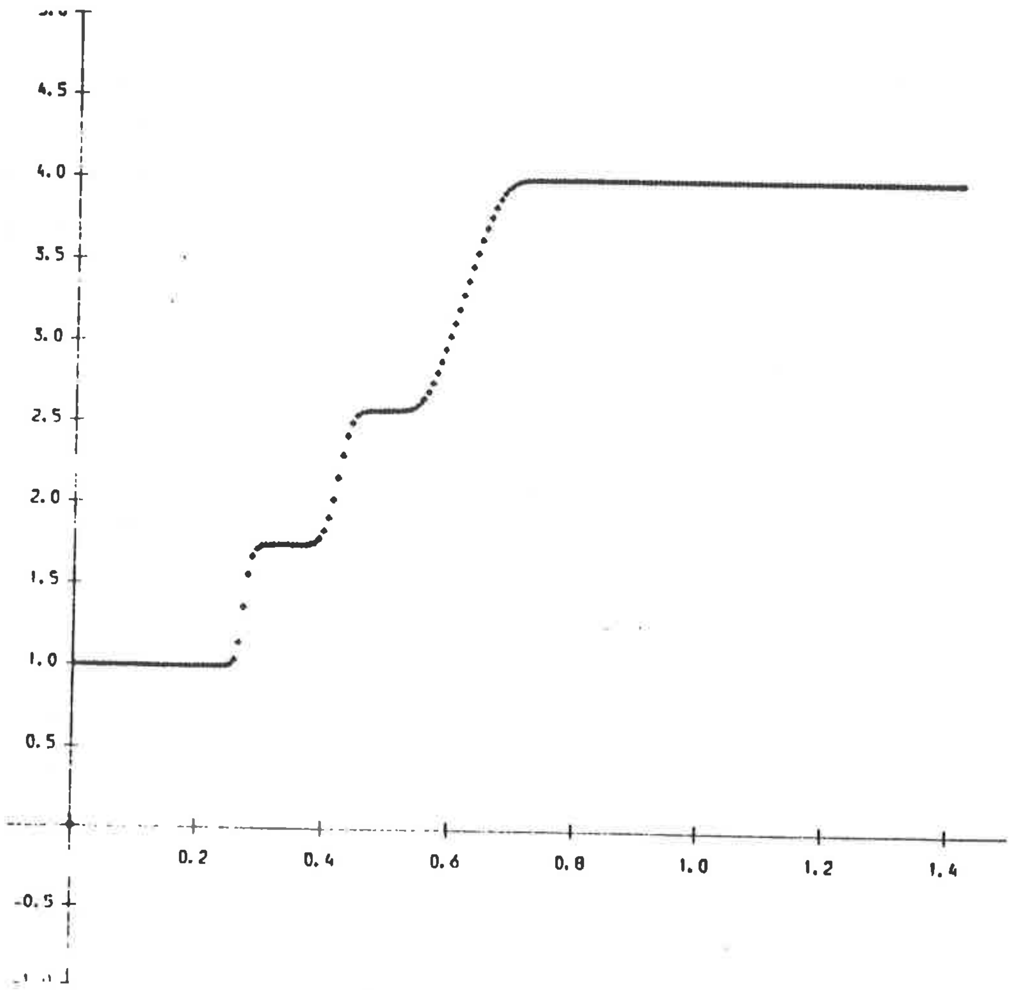
DY = 0.00500

MAX CFL : 0.25142

MAX HEIGHT : 4.00000

MIN HEIGHT : 0.00000

Figure 23



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

TIME : 0.1375

110 TIME STEPS

DT = 0.00125

DX = 0.00500

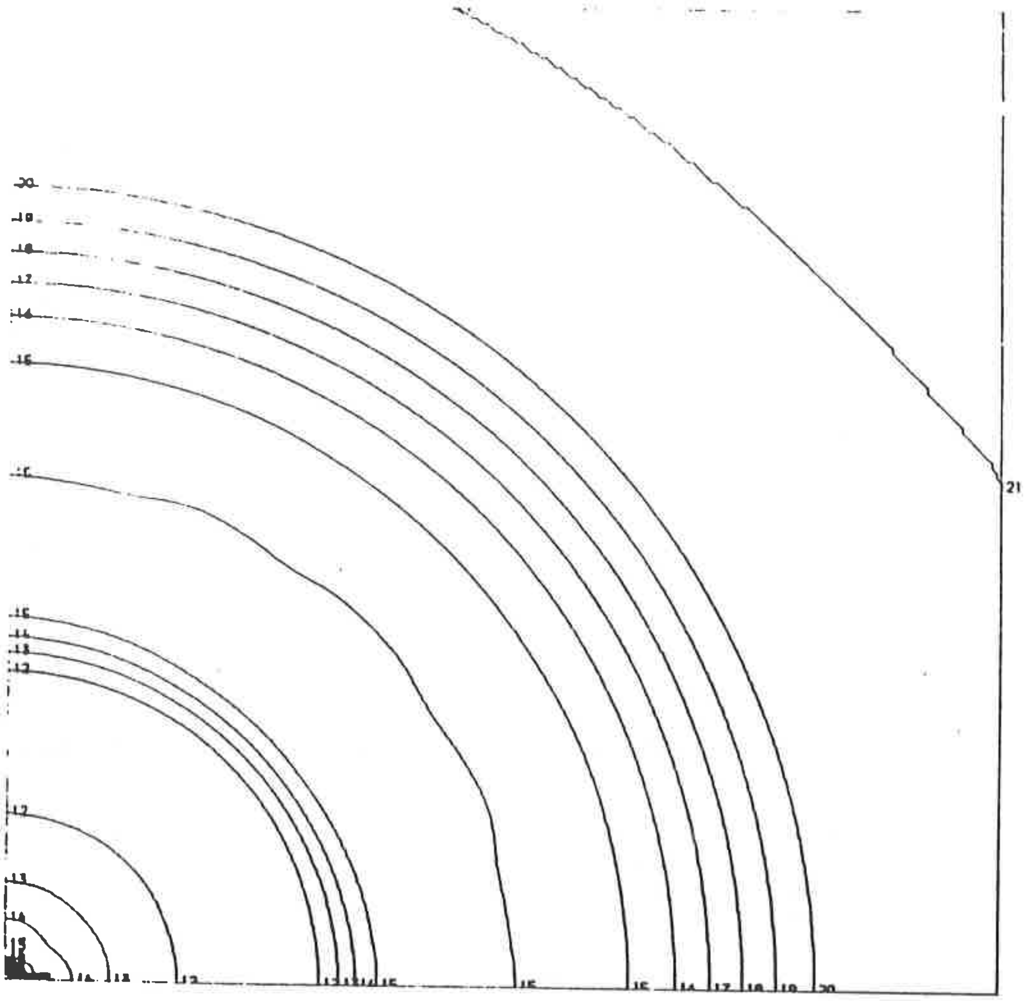
DY = 0.00500

MAX CFL : 0.25142

MAX HEIGHT : 4.00000

MIN HEIGHT : 0.00000

Figure 24



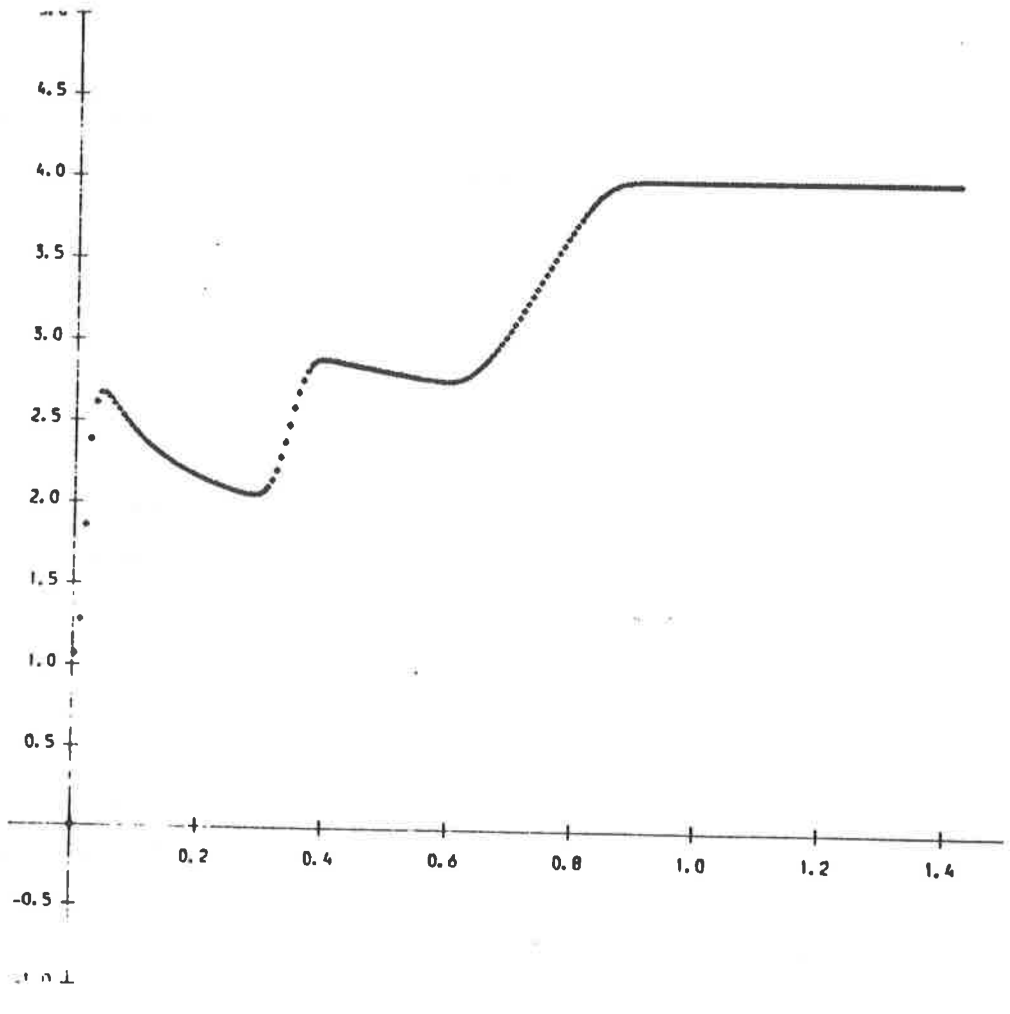
SOLUTION OF SODS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY

TIME : 0.2750
220 TIME STEPS

DT = 0.00125
DX = 0.00500
DY = 0.00500
MAX CFL : 0.32105
MAX HEIGHT : 4.00000
MIN HEIGHT : 0.00000

Figure 25



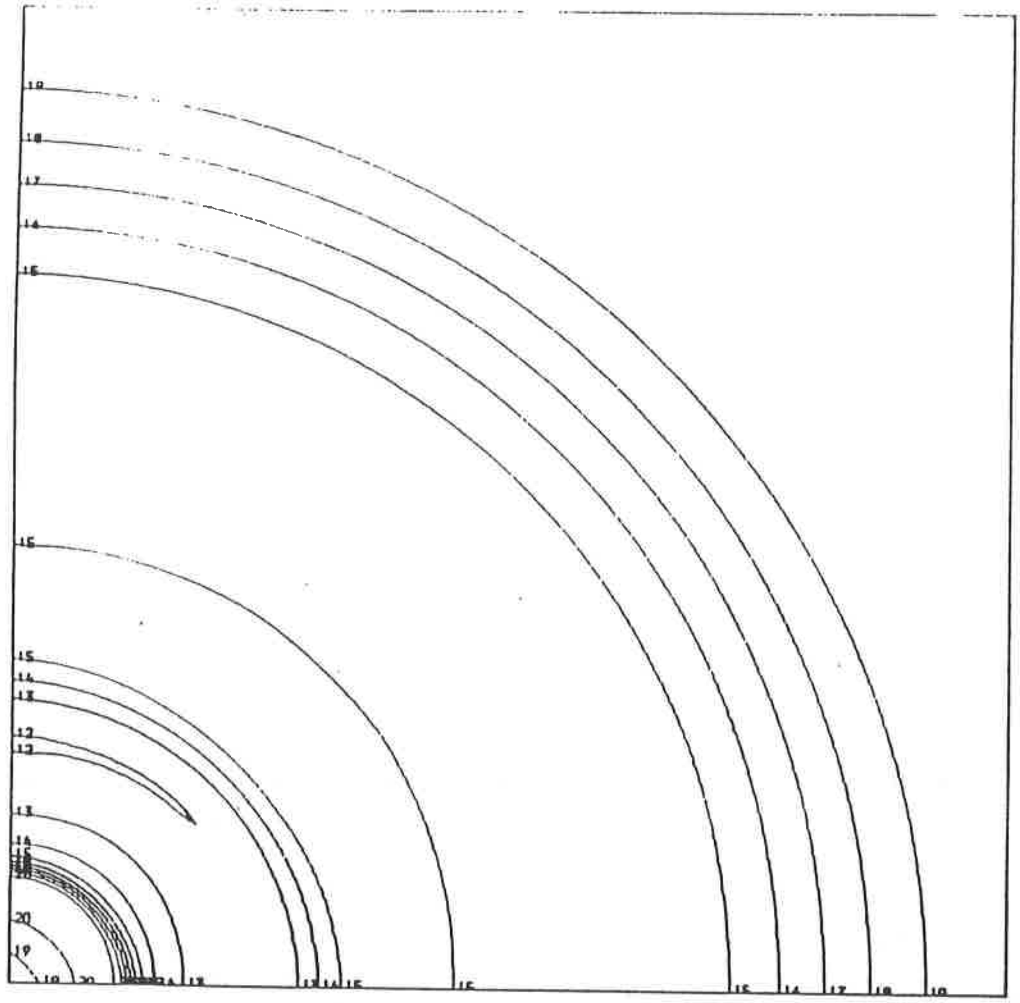
SOLUTION OF SODS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY

TIME : 0.2750
220 TIME STEPS

DT = 0.00125
DX = 0.00500
DY = 0.00500
MAX CFL : 0.32105
MAX HEIGHT : 4.00000
MIN HEIGHT : 0.00000

Figure 26

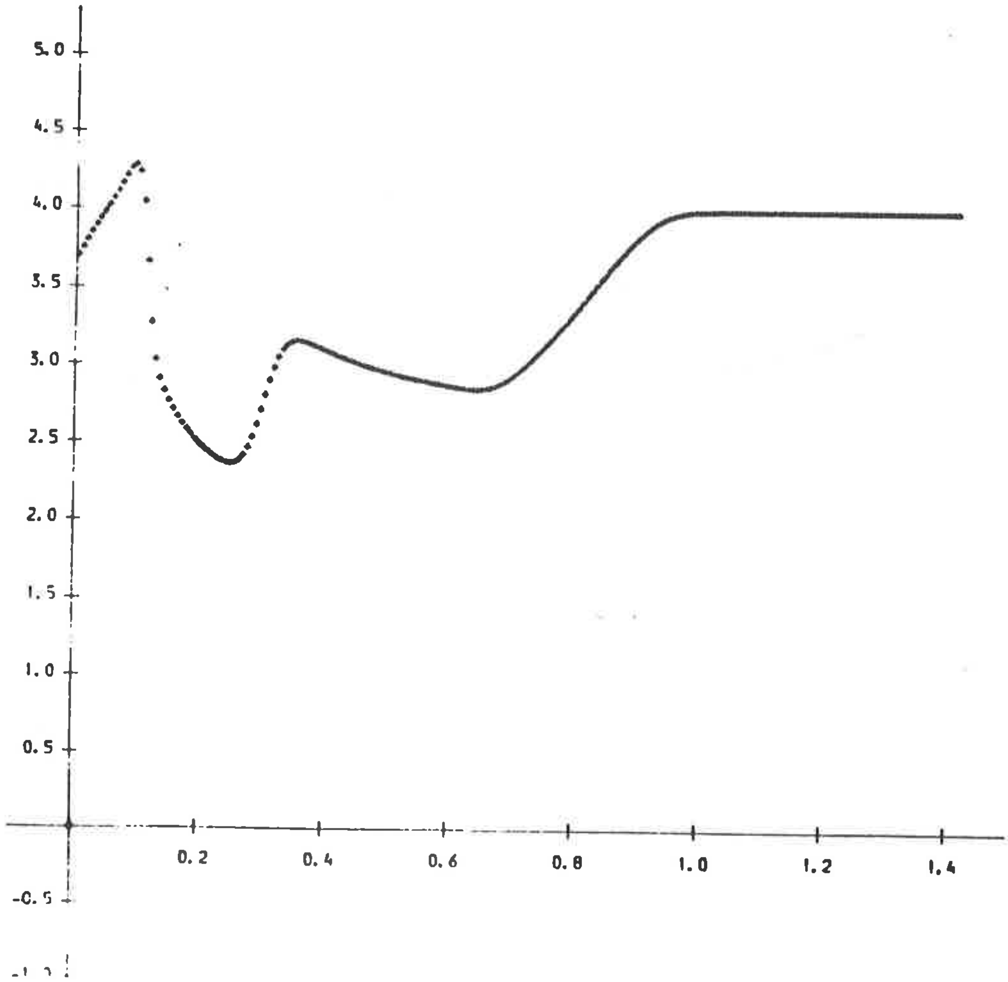


SOLUTION OF SODS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY

TIME : 0.3500
200 TIME STEPS

DT = 0.00125
DX = 0.00500
DY = 0.00500
MAX CFL : 0.33002
MAX HEIGHT : 4.29897
MIN HEIGHT : 0.00000

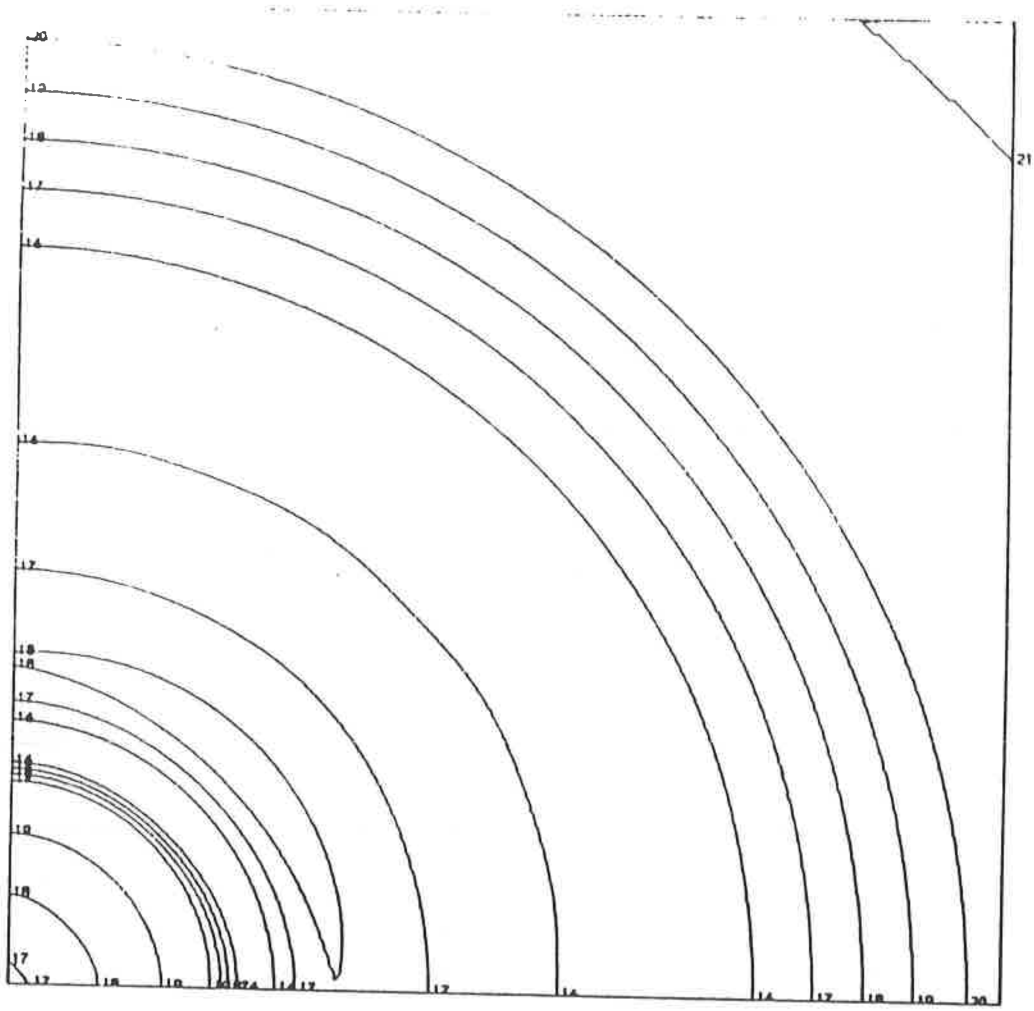


SOLUTION OF SODS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY

TIME : 0.3500
200 TIME STEPS

DT = 0.00125
DX = 0.00500
DY = 0.00500
MAX CFL : 0.33002
MAX HEIGHT : 4.29897
MIN HEIGHT : 0.00000



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

TIME : 0.4125

330 TIME STEPS

DT = 0.00125

DX = 0.00500

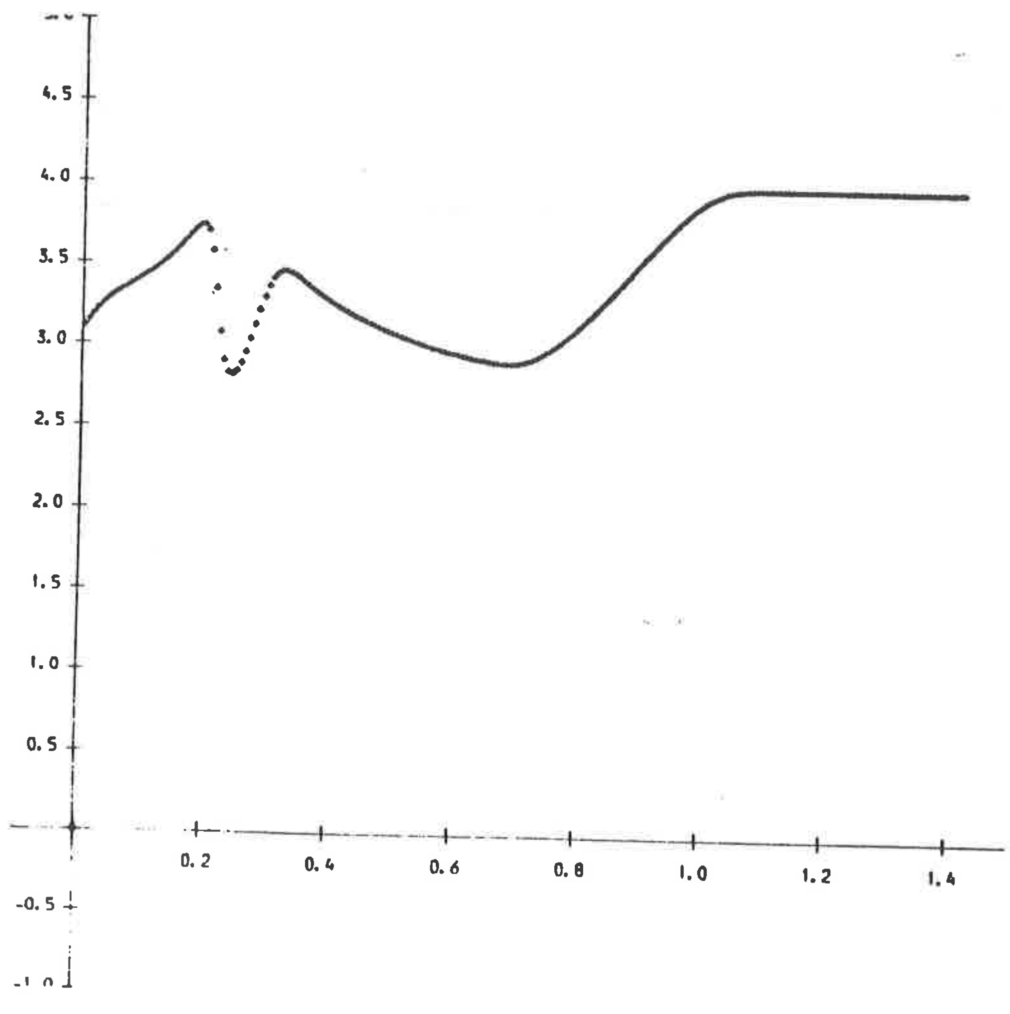
DY = 0.00500

MAX CFL : 0.33002

MAX HEIGHT : 4.00000

MIN HEIGHT : 0.00000

Figure 29



SOLUTION OF SODS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

TIME : 0.4125

330 TIME STEPS

DT = 0.00125

DX = 0.00500

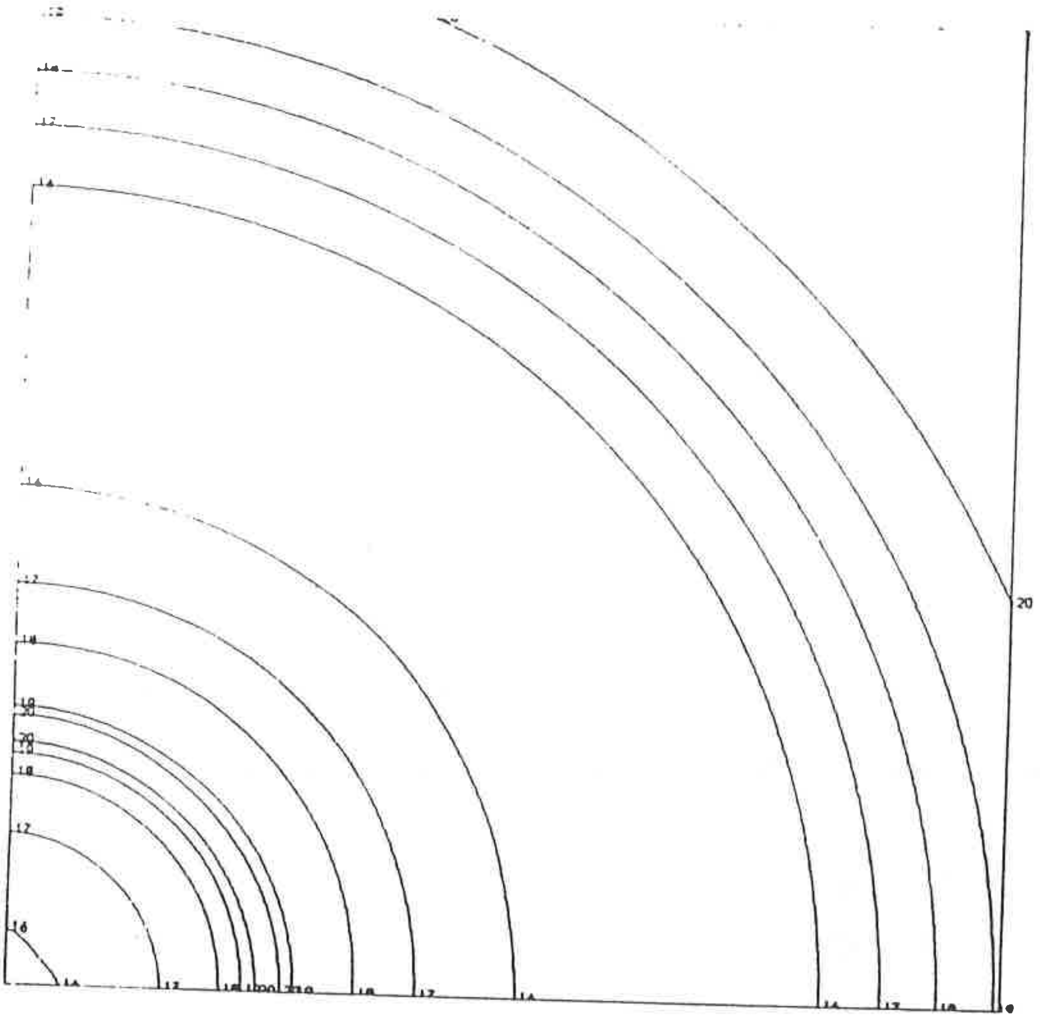
DY = 0.00500

MAX CFL : 0.33002

MAX HEIGHT : 4.00000

MIN HEIGHT : 0.00000

Figure 30



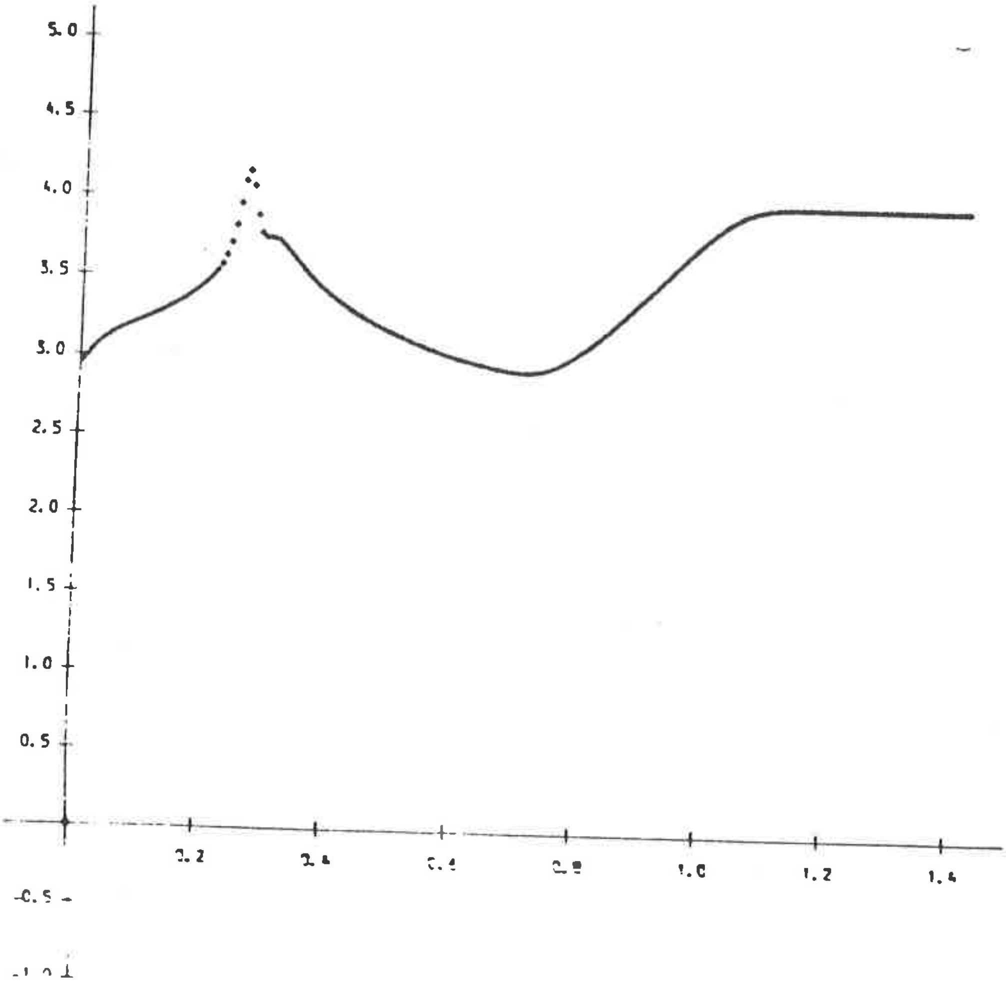
SOLUTION OF SODS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY

TIME : 0.4500
560 TIME STEPS

DT = 0.00125
DX = 0.00500
DY = 0.00500
MAX CFL : 0.33002
MAX HEIGHT : 4.17734
MIN HEIGHT : 0.00000

Figure 31



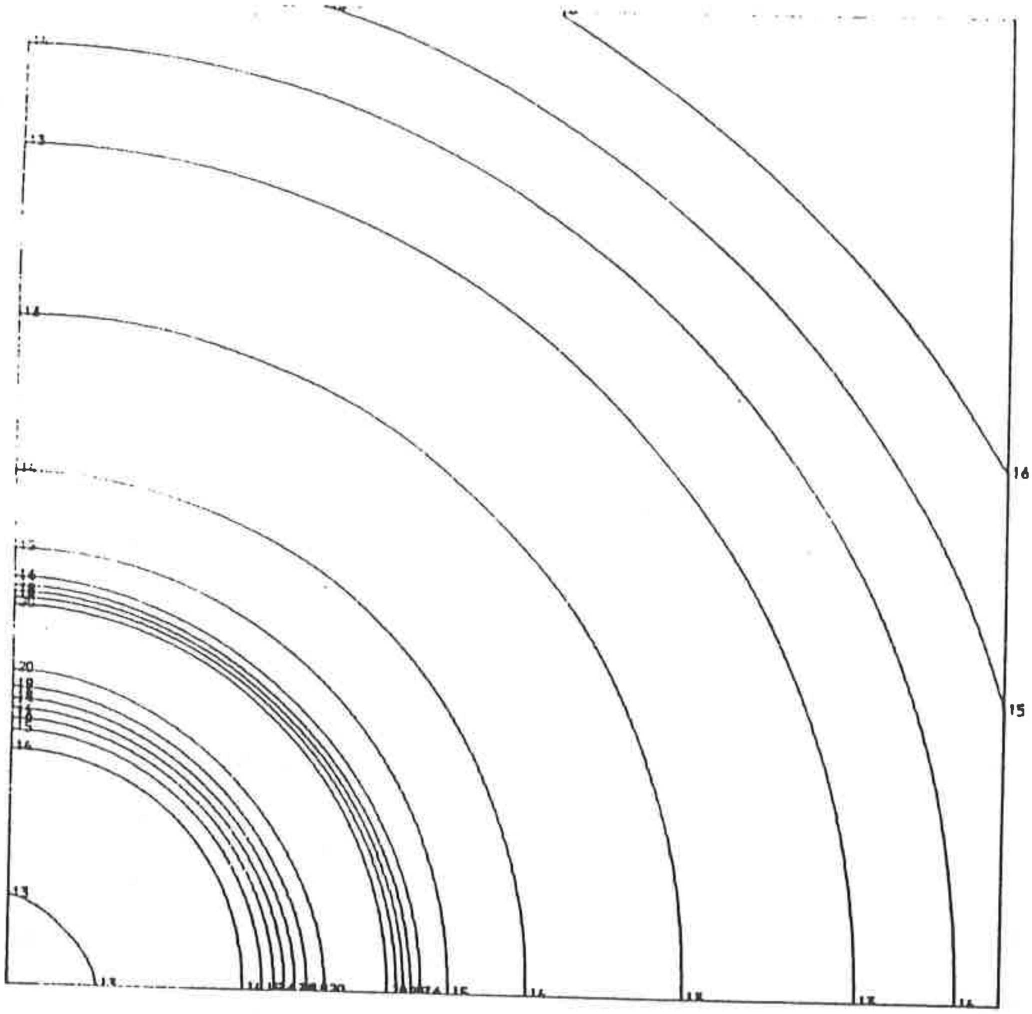
SOLUTION OF SODS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY

TIME : 0.4500
560 TIME STEPS

DT = 0.00125
DX = 0.00500
DY = 0.00500
MAX CFL : 0.33002
MAX HEIGHT : 4.17734
MIN HEIGHT : 0.00000

Figure 32



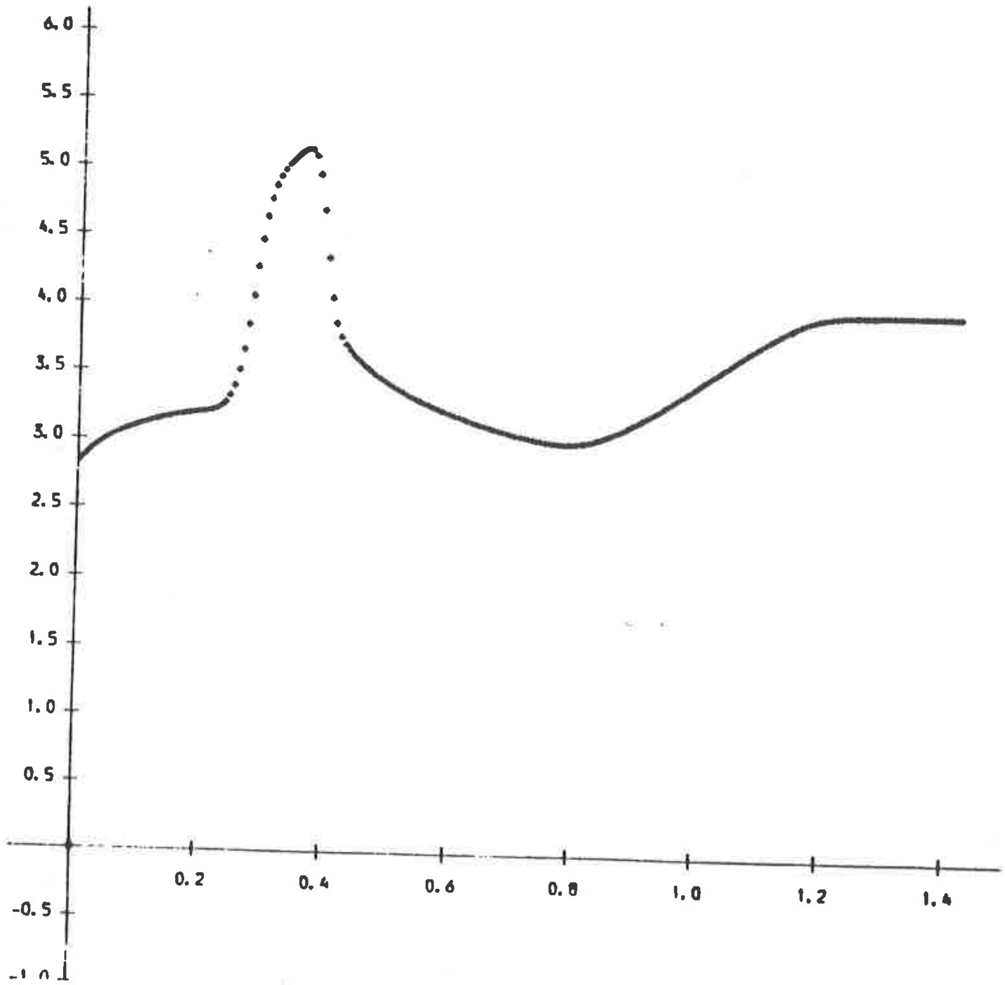
SOLUTION OF SODS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY

TIME : 0.5500
440 TIME STEPS

DT = 0.00125
DX = 0.00500
DY = 0.00500
MAX CFL : 0.33002
MAX HEIGHT : 5.14414
MIN HEIGHT : 0.00000

Figure 33



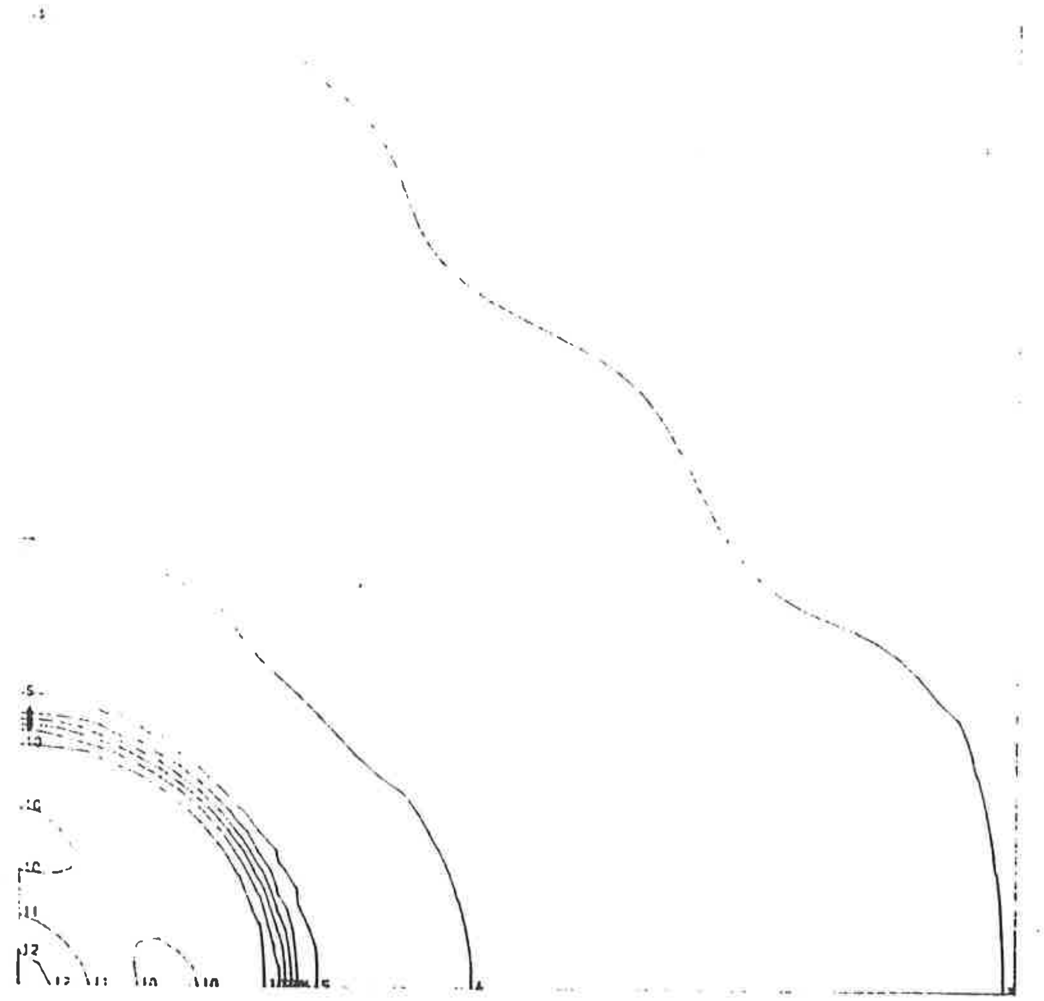
SOLUTION OF SODS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY

TIME : 0.5500
440 TIME STEPS

DT = 0.00125
DX = 0.00500
DY = 0.00500
MAX CFL : 0.33002
MAX HEIGHT : 5.14414
MIN HEIGHT : 0.00000

Figure 34

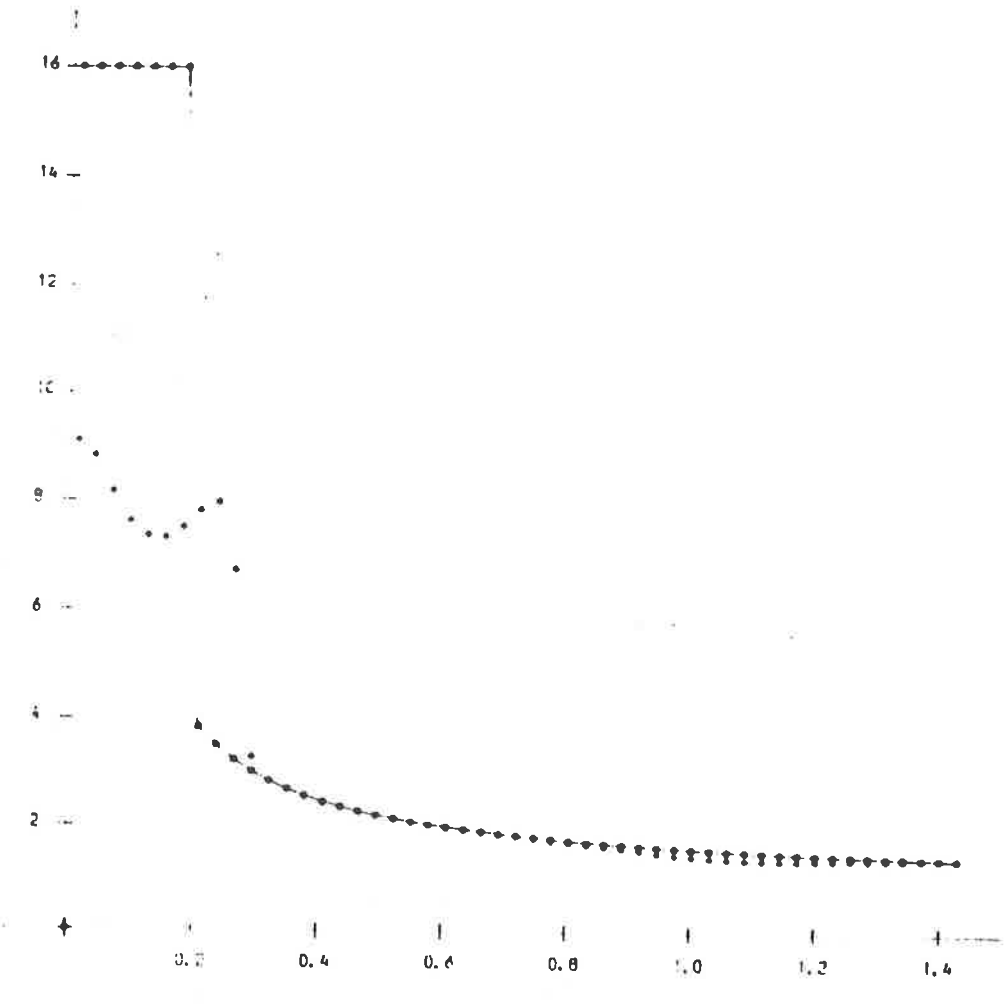


SOLUTION OF NOHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY
AVERAGE ERROR : 0.42604
MAX ERROR : 8.97024
TIME : 0.6000
84 TIME STEPS

DT = 0.00500
DX = 0.02000
DT = 0.02000
MAX CFL : 0.14591
MAX HEIGHT : 16.00000
MIN HEIGHT : 0.00000

Figure 35

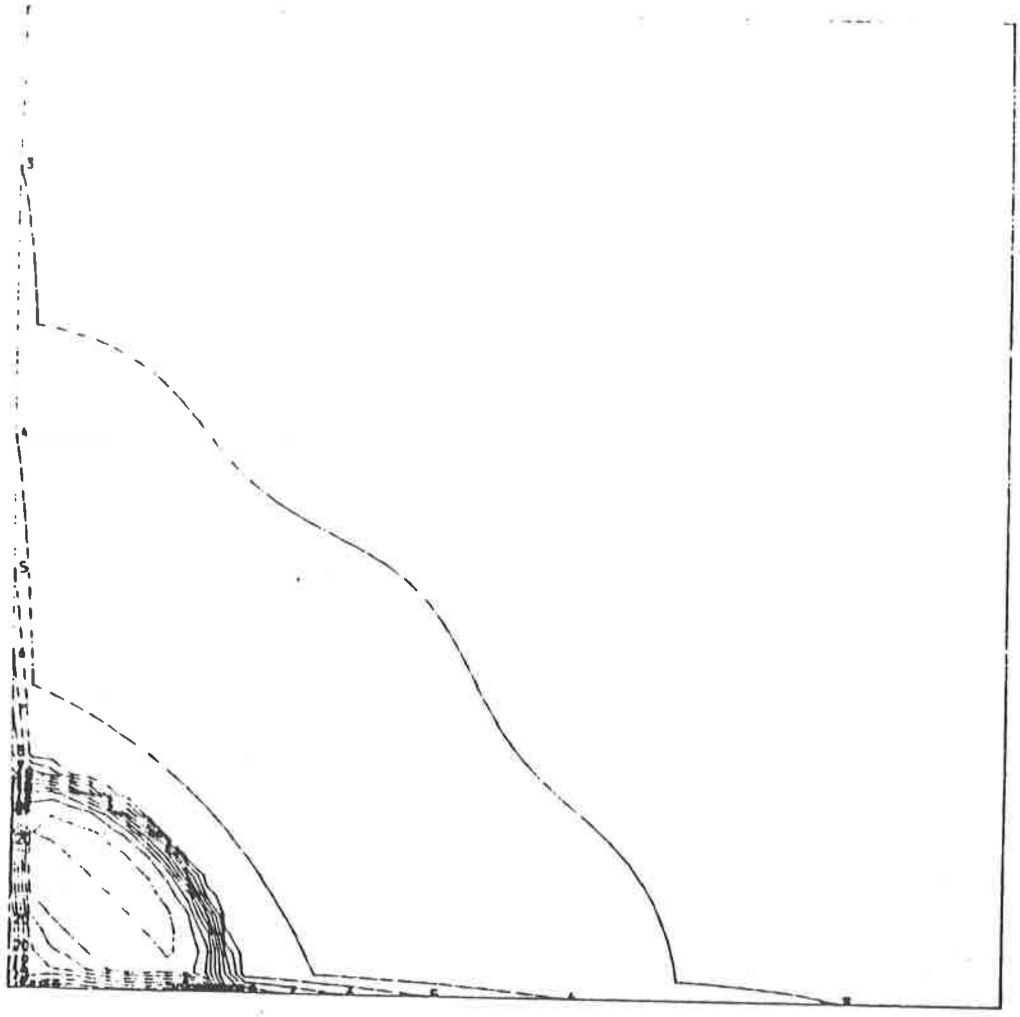


SOLUTION OF NOHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY
AVERAGE ERROR : 0.42604
MAX ERROR : 8.97024
TIME : 0.6000
84 TIME STEPS

DT = 0.00500
DX = 0.02000
DT = 0.02000
MAX CFL : 0.14591
MAX HEIGHT : 16.00000
MIN HEIGHT : 0.00000

Figure 36



SOLUTION OF NOHS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

AVERAGE ERROR : 0.28002

MAX ERROR : 10.08361

TIME : 0.6000

120 TIME STEPS

DT = 0.00500

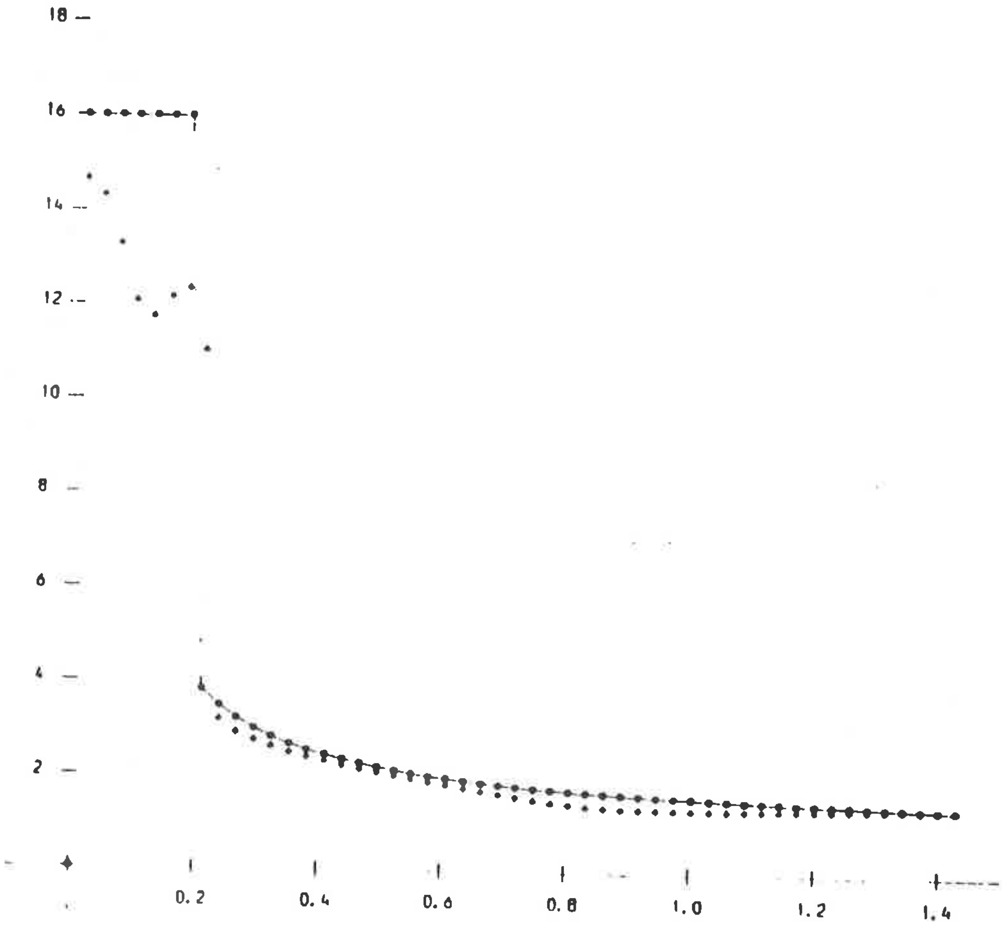
DX = 0.02000

DY = 0.02000

MAX CFL : 0.15147

MAX HEIGHT : 18.71783

MIN HEIGHT : 0.00000



SOLUTION OF NOHS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

AVERAGE ERROR : 0.28002

MAX ERROR : 10.08361

TIME : 0.6000

120 TIME STEPS

DT = 0.00500

DX = 0.02000

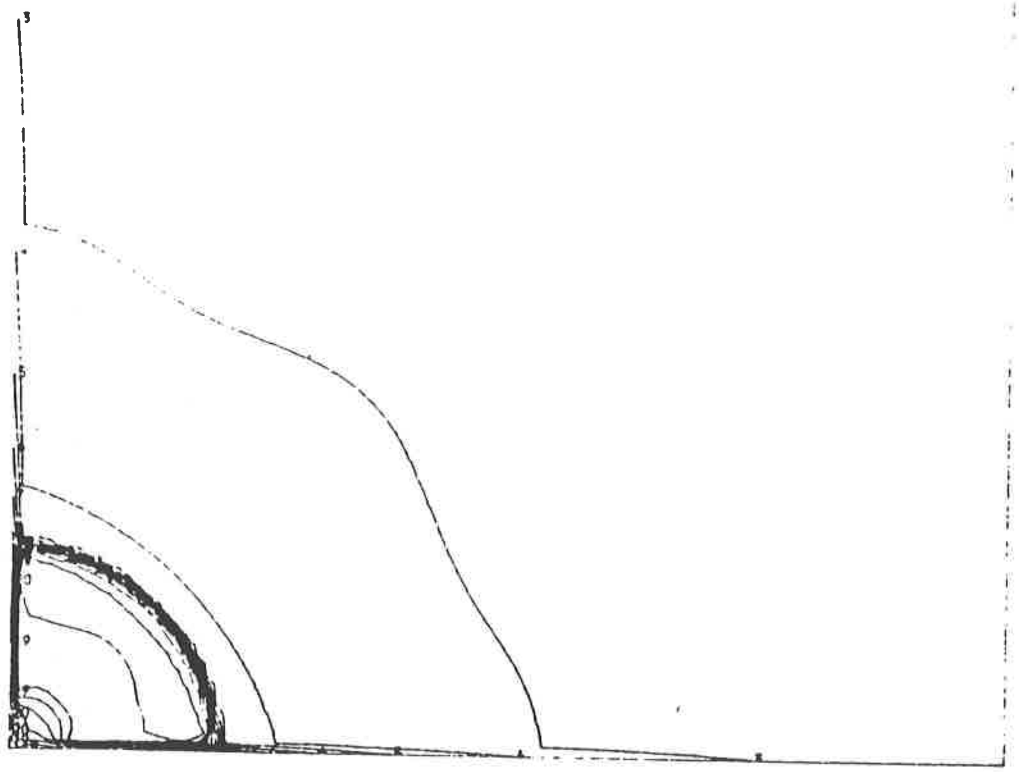
DY = 0.02000

MAX CFL : 0.15147

MAX HEIGHT : 18.71783

MIN HEIGHT : 0.00000

Figure 2

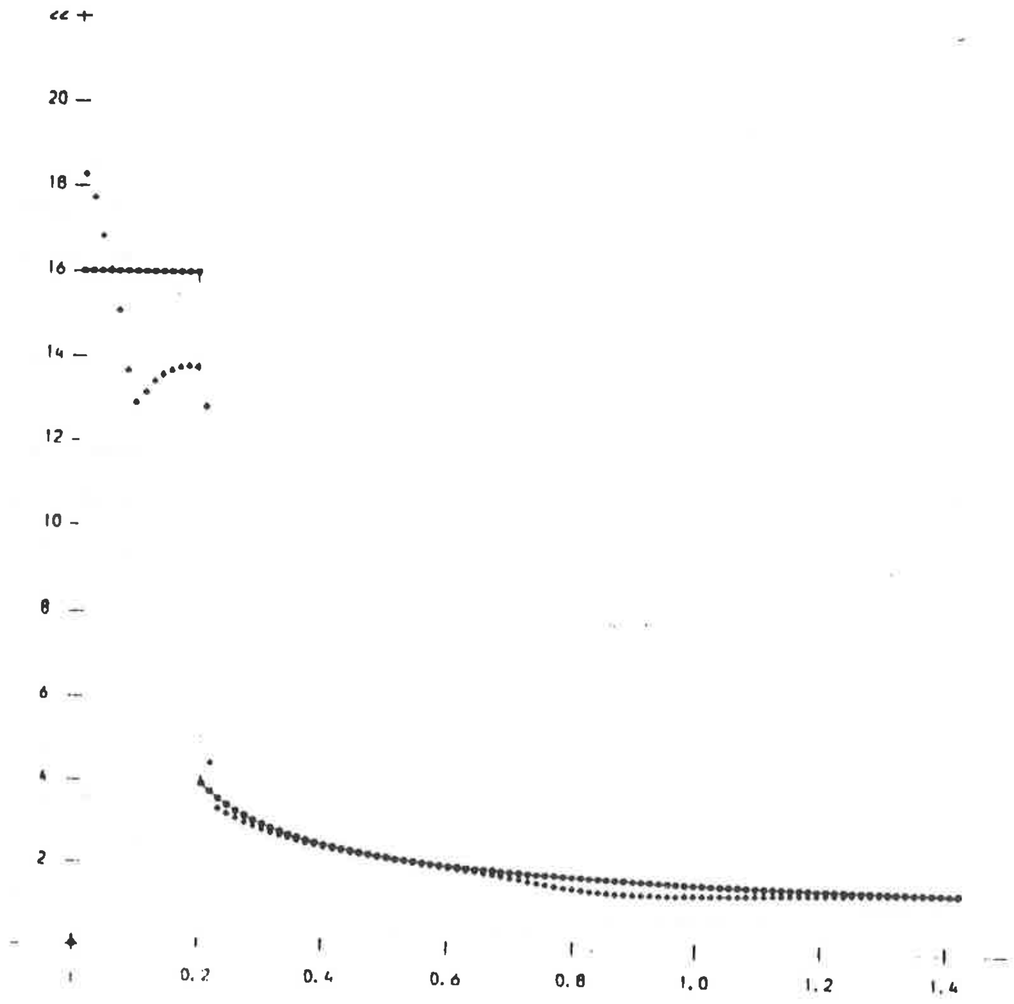


SOLUTION OF NOHS PROBLEM
USING [2] OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY
AVERAGE ERROR : 0.20782
MAX ERROR : 12.17207
TIME : 0.6000
240 TIME STEPS

DT = 0.00250
DX = 0.01000
DY = 0.01000
MAX CFL : 0.15191
MAX HEIGHT : 21.10251
MIN HEIGHT : 0.00000

Figure 3



SOLUTION OF NOHS PROBLEM
USING [2] OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY
AVERAGE ERROR : 0.20782
MAX ERROR : 12.17207
TIME : 0.6000
240 TIME STEPS

DT = 0.00250
DX = 0.01000
DY = 0.01000
MAX CFL : 0.15191
MAX HEIGHT : 21.10251
MIN HEIGHT : 0.00000

Figure 4

SOLUTION OF NOHS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

AVERAGE ERROR : 0.02083

MAX ERROR : 10.08614

TIME : 0.1500

120 TIME STEPS

DT = 0.00125

DX = 0.00500

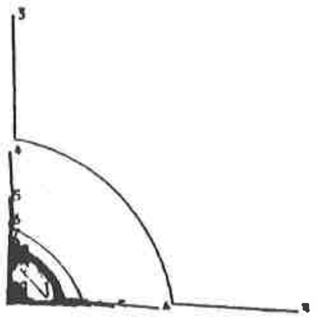
DY = 0.00500

MAX CFL : 0.15147

MAX HEIGHT : 18.71971

MIN HEIGHT : 0.00000

Figure 5



SOLUTION OF NOHS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

AVERAGE ERROR : 0.02083

MAX ERROR : 10.08614

TIME : 0.1500

120 TIME STEPS

DT = 0.00125

DX = 0.00500

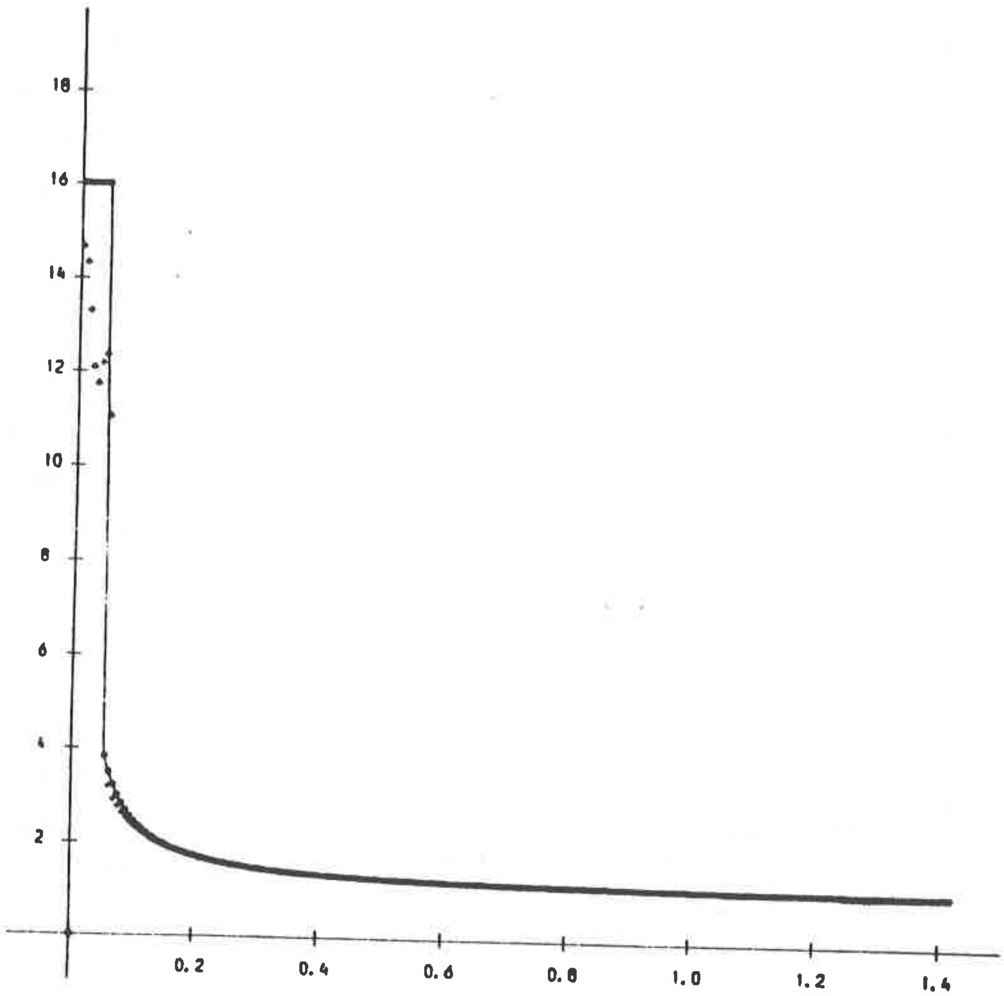
DY = 0.00500

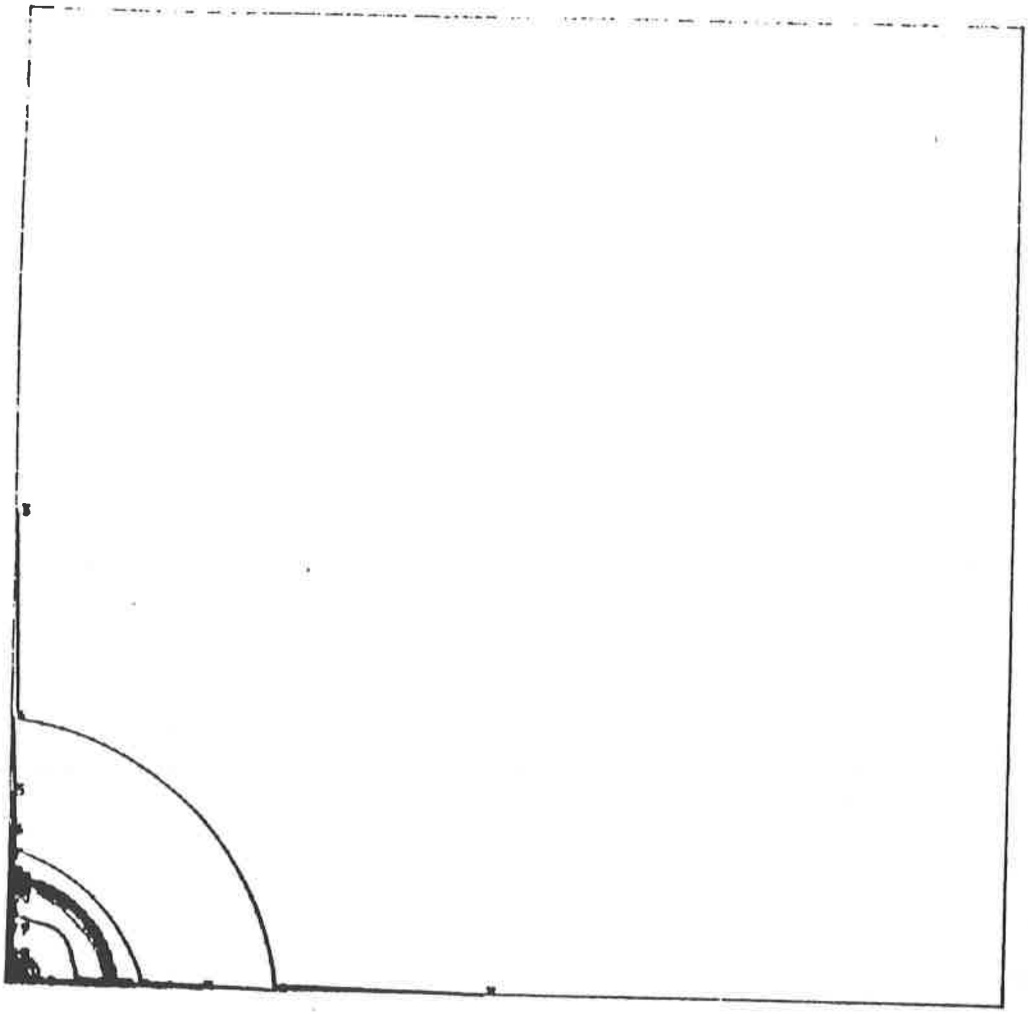
MAX CFL : 0.15147

MAX HEIGHT : 18.71971

MIN HEIGHT : 0.00000

Figure 6

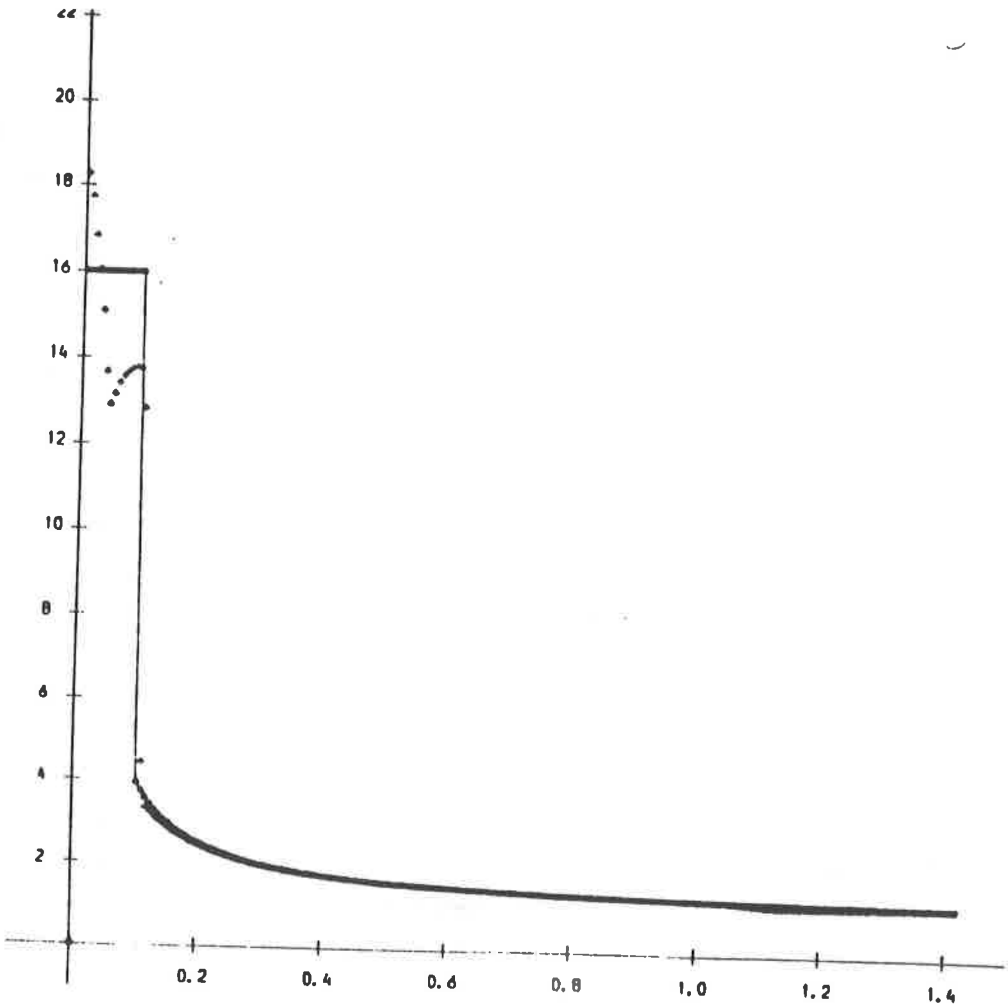




SOLUTION OF NOHS PROBLEM
USING L21 OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY
AVERAGE ERROR : 0.05515
MAX ERROR : 12.17227
TIME : 0.3000
240 TIME STEPS

DT = 0.00125
DX = 0.00500
DY = 0.00500
MAX CFL : 0.15191
MAX HEIGHT : 21.10243
MIN HEIGHT : 0.00000

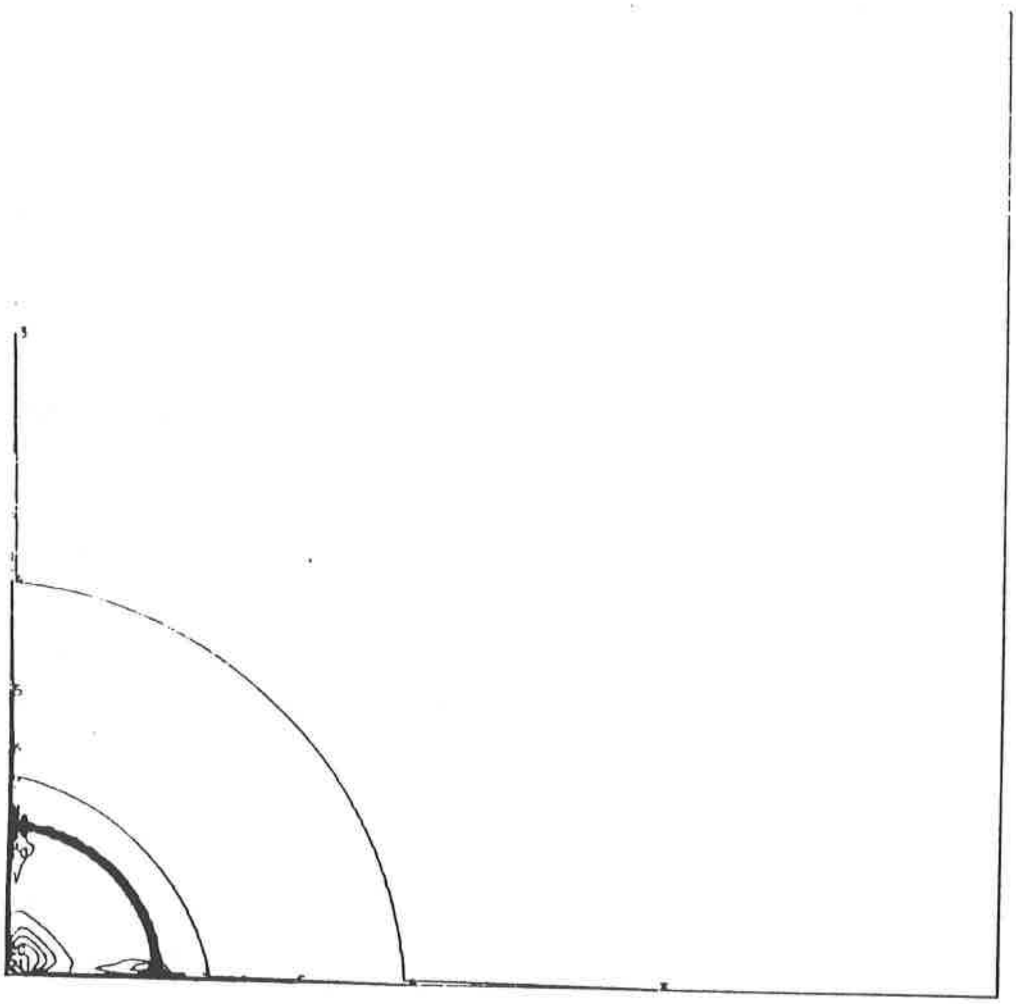


SOLUTION OF NOHS PROBLEM
USING L21 OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY
AVERAGE ERROR : 0.05515
MAX ERROR : 12.17227
TIME : 0.3000
240 TIME STEPS

DT = 0.00125
DX = 0.00500
DY = 0.00500
MAX CFL : 0.15191
MAX HEIGHT : 21.10243
MIN HEIGHT : 0.00000

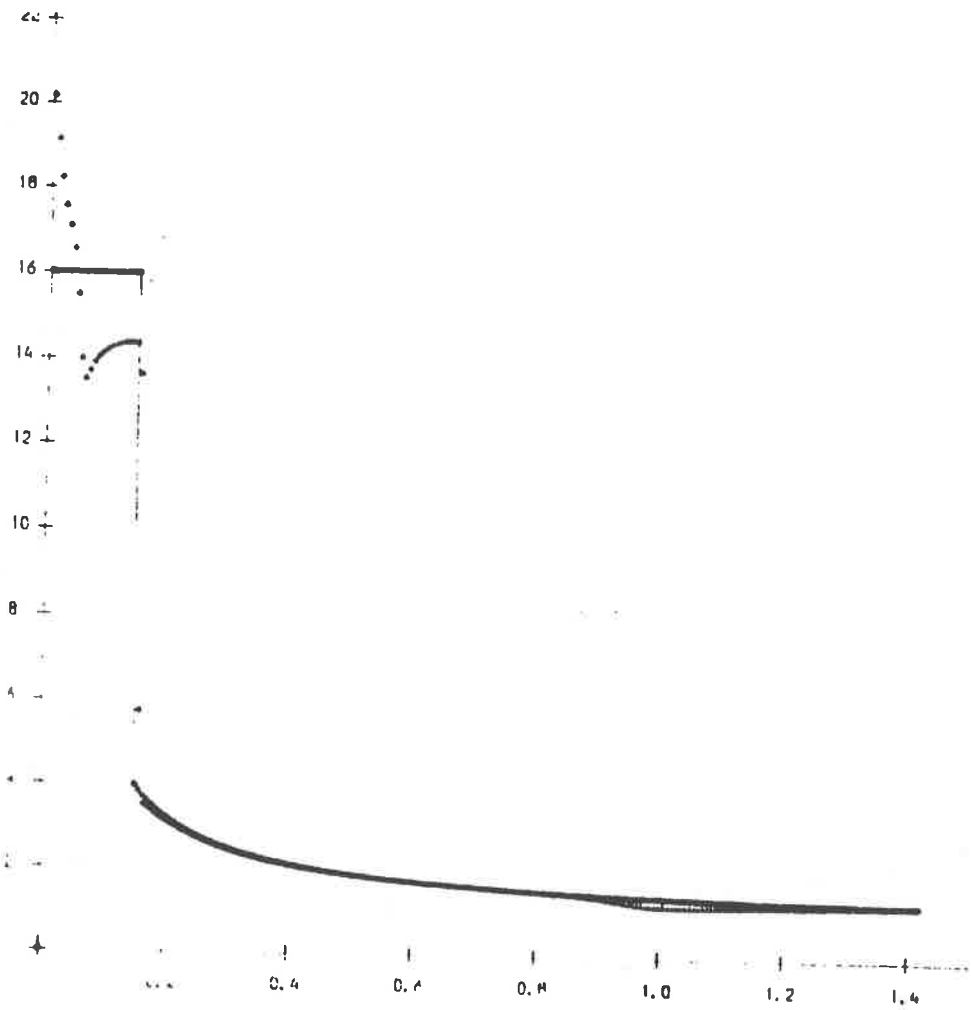
Figure 8



SOLUTION OF NOHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY
AVERAGE ERROR : 0.09689
MAX ERROR : 11.18870
TIME : 0.4500
560 TIME STEPS

DT = 0.00125
DX = 0.00500
DY = 0.00500
MAX CFL : 0.15835
MAX HEIGHT : 21.10834
MIN HEIGHT : 0.00000

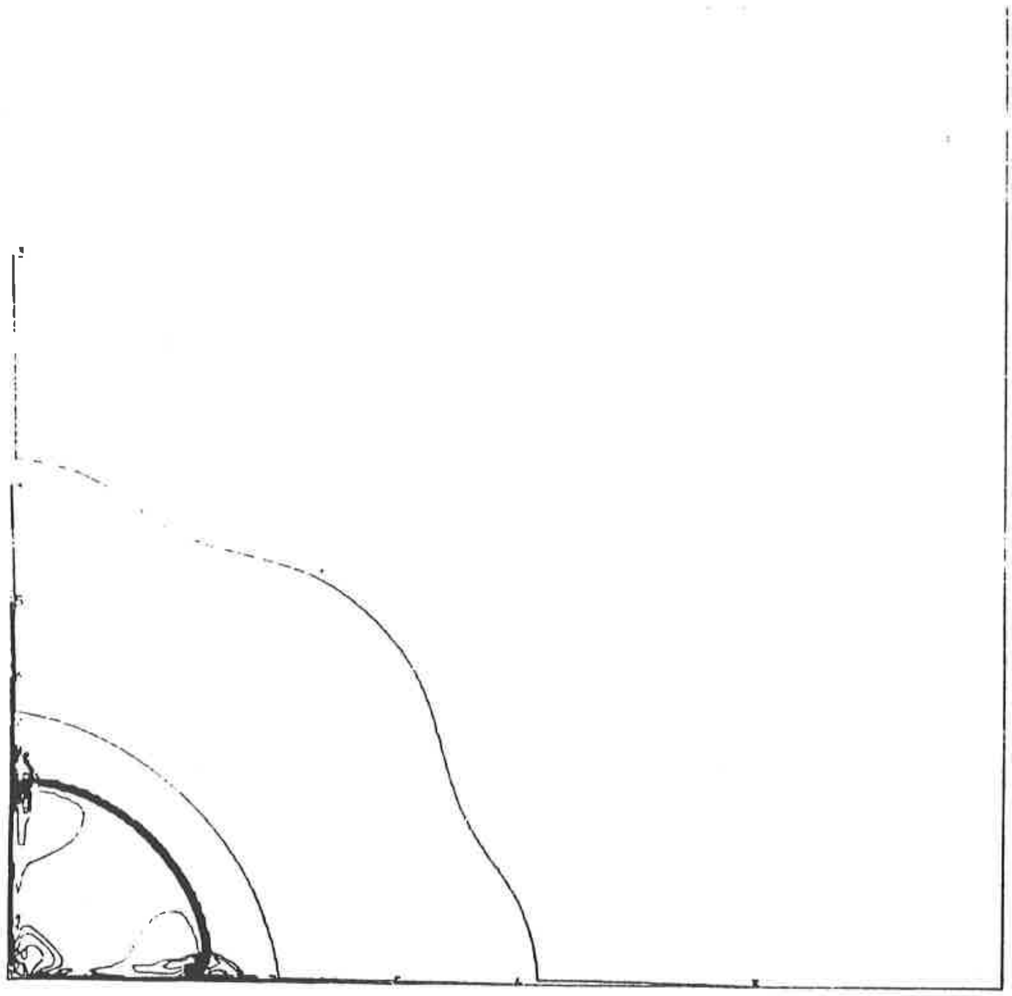


SOLUTION OF NOHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY
AVERAGE ERROR : 0.09689
MAX ERROR : 11.18870
TIME : 0.4500
560 TIME STEPS

DT = 0.00125
DX = 0.00500
DY = 0.00500
MAX CFL : 0.15835
MAX HEIGHT : 21.10834
MIN HEIGHT : 0.00000

figure 10



SOLUTION OF NOHS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

AVERAGE ERROR : 0.14589

MAX ERROR : 10.67271

TIME : 0.6000

480 TIME STEPS

DT = 0.00125

DX = 0.00500

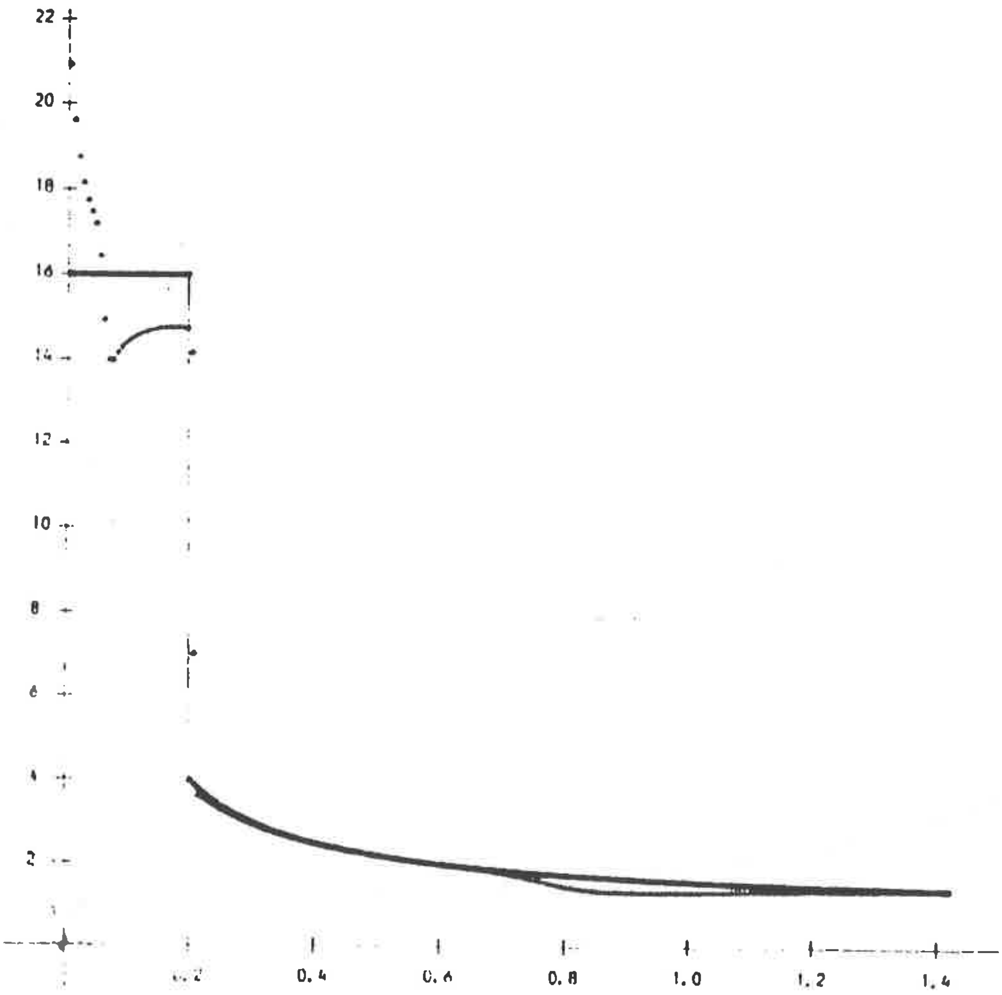
DY = 0.00500

MAX CFL : 0.20259

MAX HEIGHT : 21.22532

MIN HEIGHT : 0.00000

Figure 11



SOLUTION OF NOHS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

AVERAGE ERROR : 0.14589

MAX ERROR : 10.67271

TIME : 0.6000

480 TIME STEPS

DT = 0.00125

DX = 0.00500

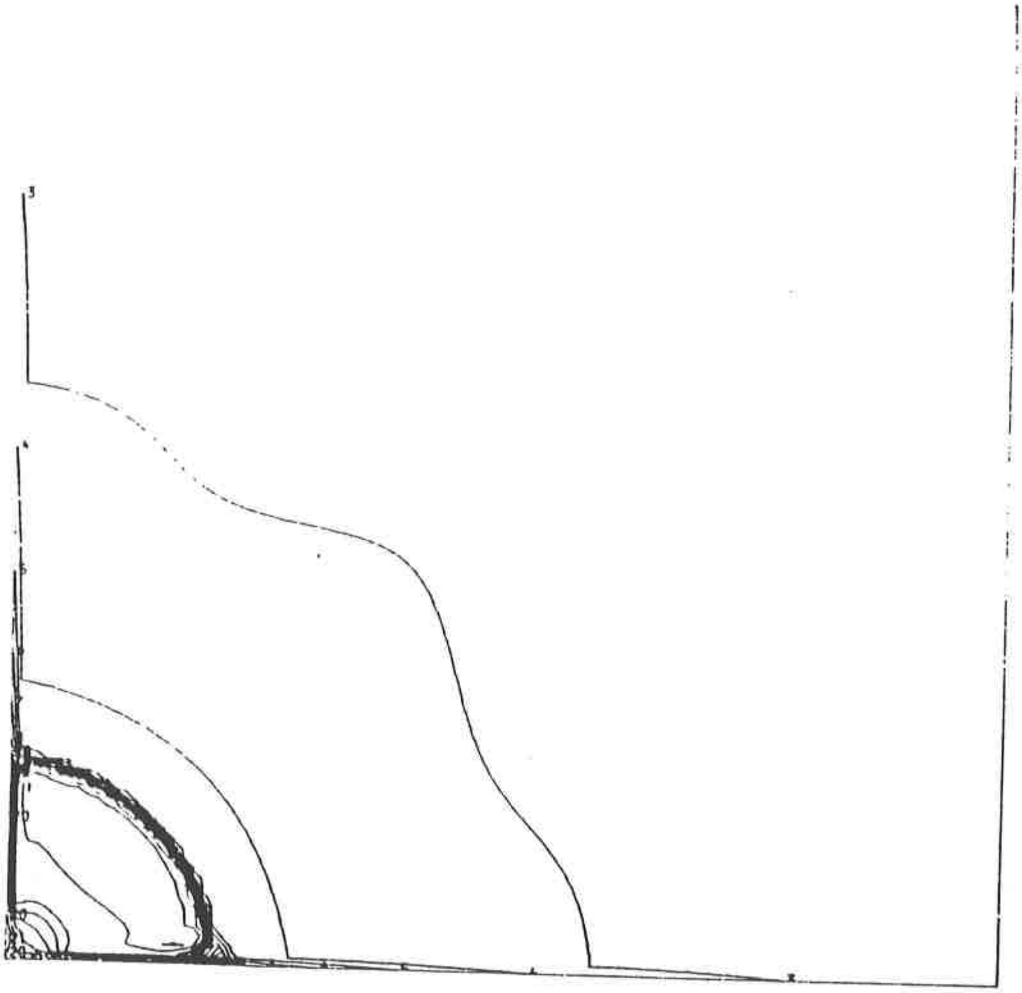
DY = 0.00500

MAX CFL : 0.20259

MAX HEIGHT : 21.22532

MIN HEIGHT : 0.00000

Figure 12



SOLUTION OF NOHS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

AVERAGE ERROR : 0.18853

MAX ERROR : 11.04036

TIME : 0.6000

100 TIME STEPS

DT = 0.00600

DX = 0.01000

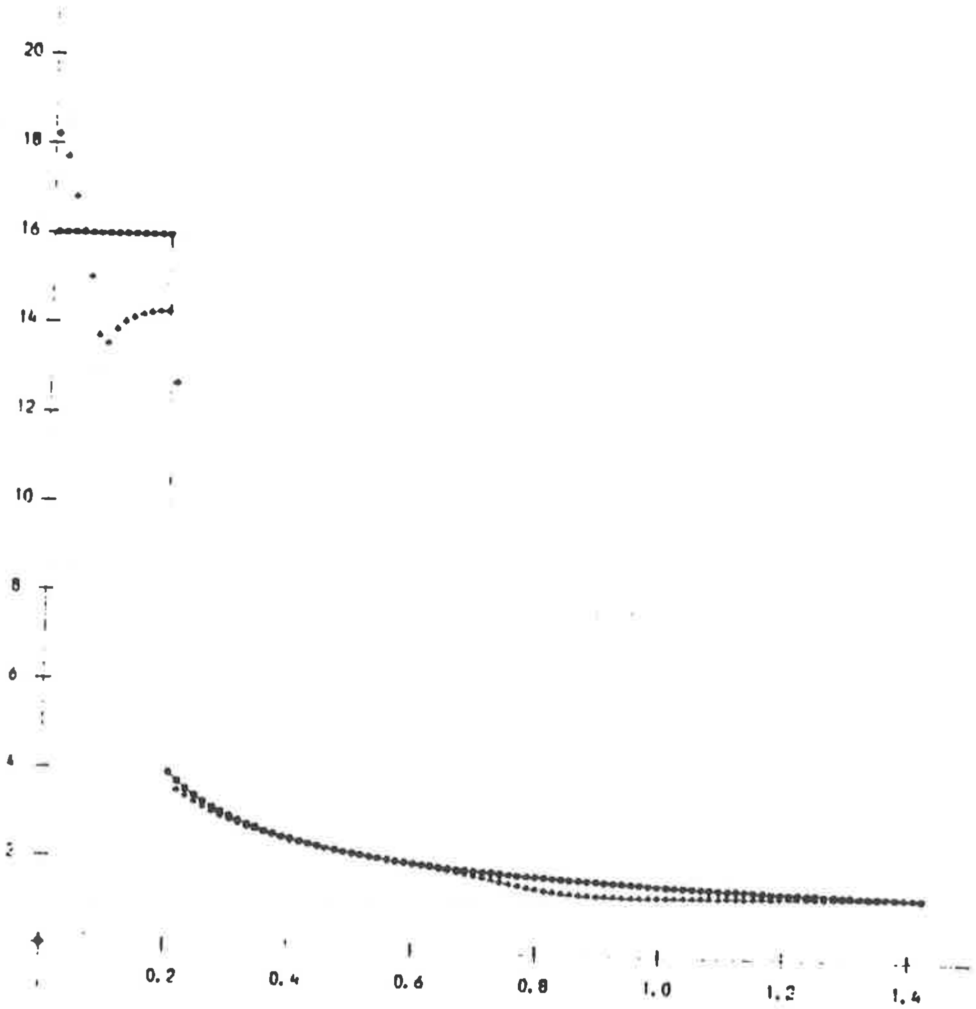
DY = 0.01000

MAX CFL : 0.39479

MAX HEIGHT : 19.92585

MIN HEIGHT : 0.00000

Figure 13



SOLUTION OF NOHS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

DENSITY

AVERAGE ERROR : 0.18853

MAX ERROR : 11.04036

TIME : 0.6000

100 TIME STEPS

DT = 0.00600

DX = 0.01000

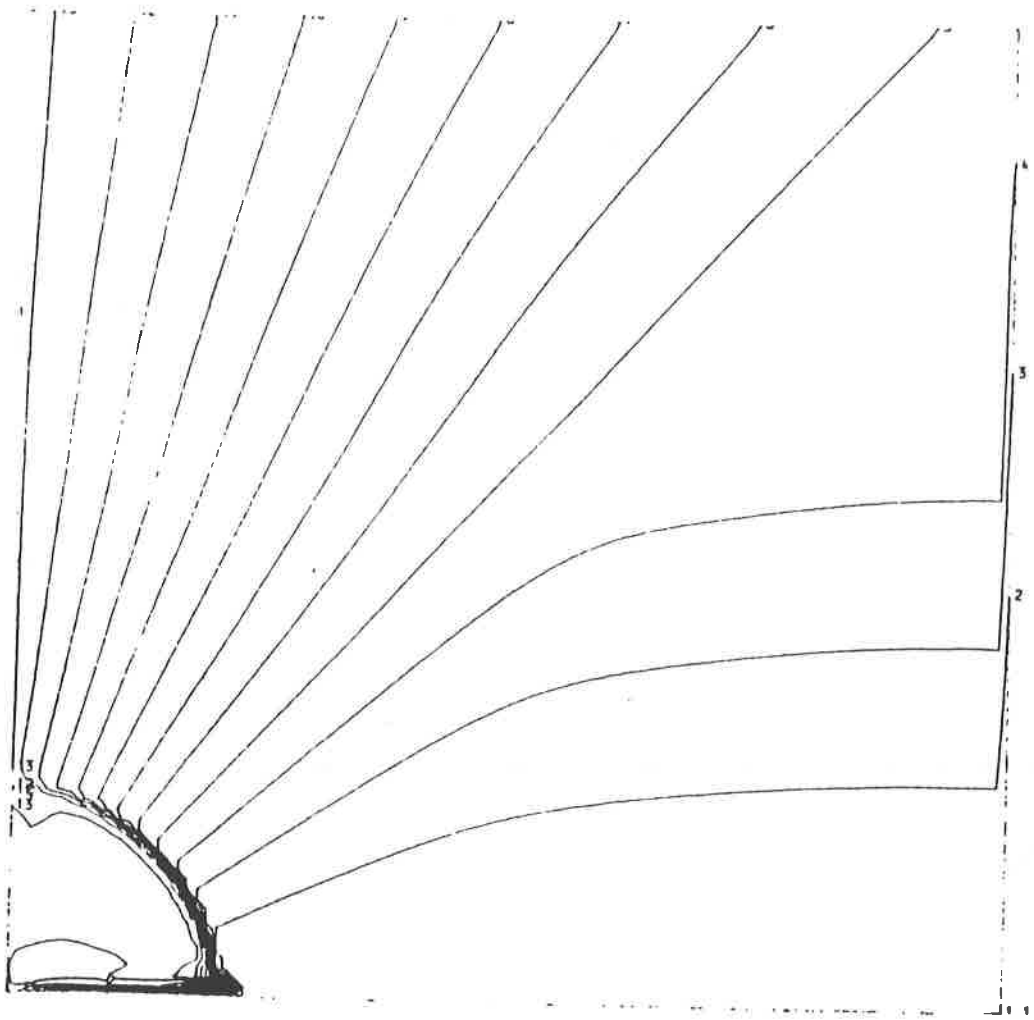
DY = 0.01000

MAX CFL : 0.39479

MAX HEIGHT : 19.92585

MIN HEIGHT : 0.00000

Figure 14

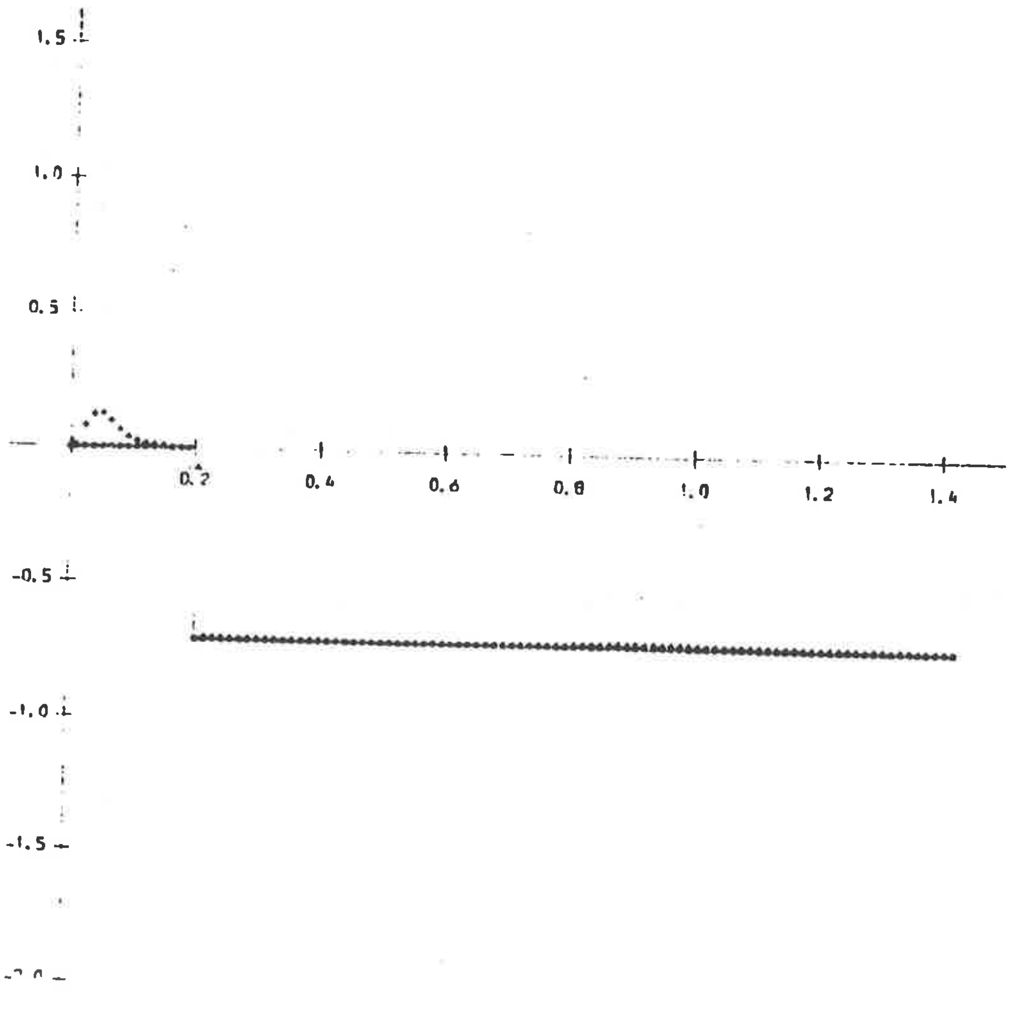


SOLUTION OF NCHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
X-VELOCITY
AVERAGE ERROR = 0.20858
MAX ERROR = 1.32259
TIME = 0.6000
100 TIME STEPS

DT = 0.00600
DX = 0.01000
DY = 0.01000
MAX CFL = 0.39479
MAX HEIGHT = 0.61548
MIN HEIGHT = -0.99999

Figure 15

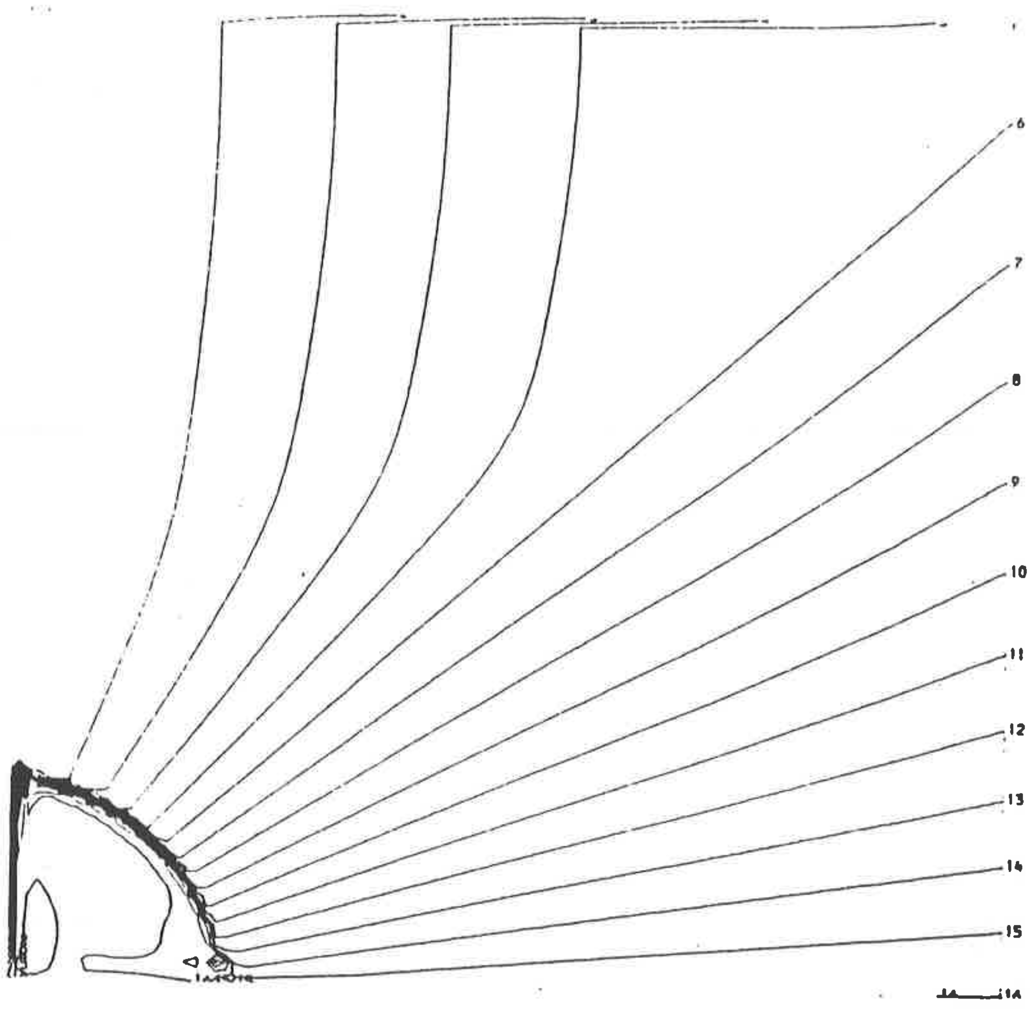


SOLUTION OF NCHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
X-VELOCITY
AVERAGE ERROR = 0.20858
MAX ERROR = 1.32259
TIME = 0.6000
100 TIME STEPS

DT = 0.00600
DX = 0.01000
DY = 0.01000
MAX CFL = 0.39479
MAX HEIGHT = 0.61548
MIN HEIGHT = -0.99999

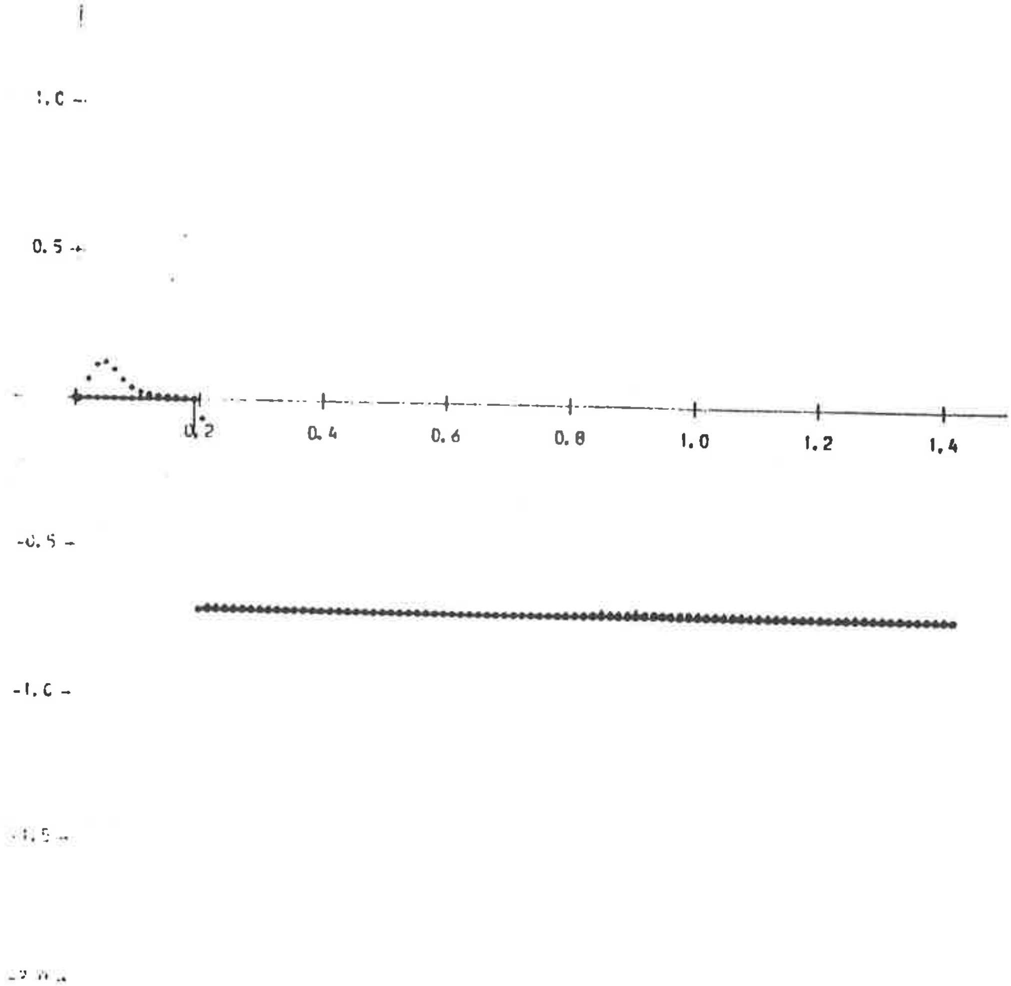
Figure 16



SOLUTION OF NOHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
Y-VELOCITY
AVERAGE ERROR . 0.20841
MAX ERROR . 1.0330-
TIME . 0.6000
100 TIME STEPS

DT . 0.00600
DX . 0.01000
DY . 0.01000
MAX CFL . 0.39479
MAX HEIGHT . 0.32593
MIN HEIGHT . -0.99999

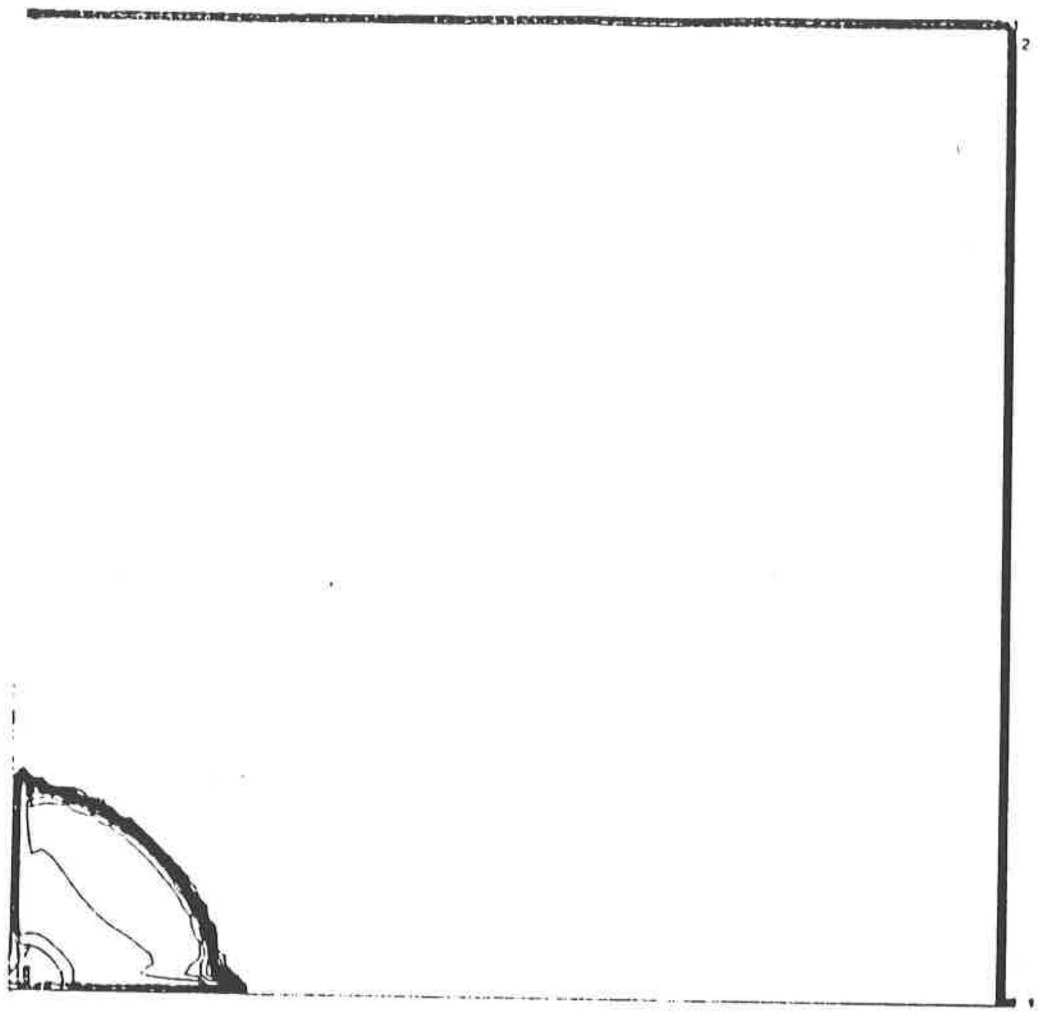


SOLUTION OF NOHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
Y-VELOCITY
AVERAGE ERROR . 0.20841
MAX ERROR . 1.03304
TIME . 0.6000
100 TIME STEPS

DT . 0.00600
DX . 0.01000
DY . 0.01000
MAX CFL . 0.39479
MAX HEIGHT . 0.32593
MIN HEIGHT . -0.99999

Figure 18

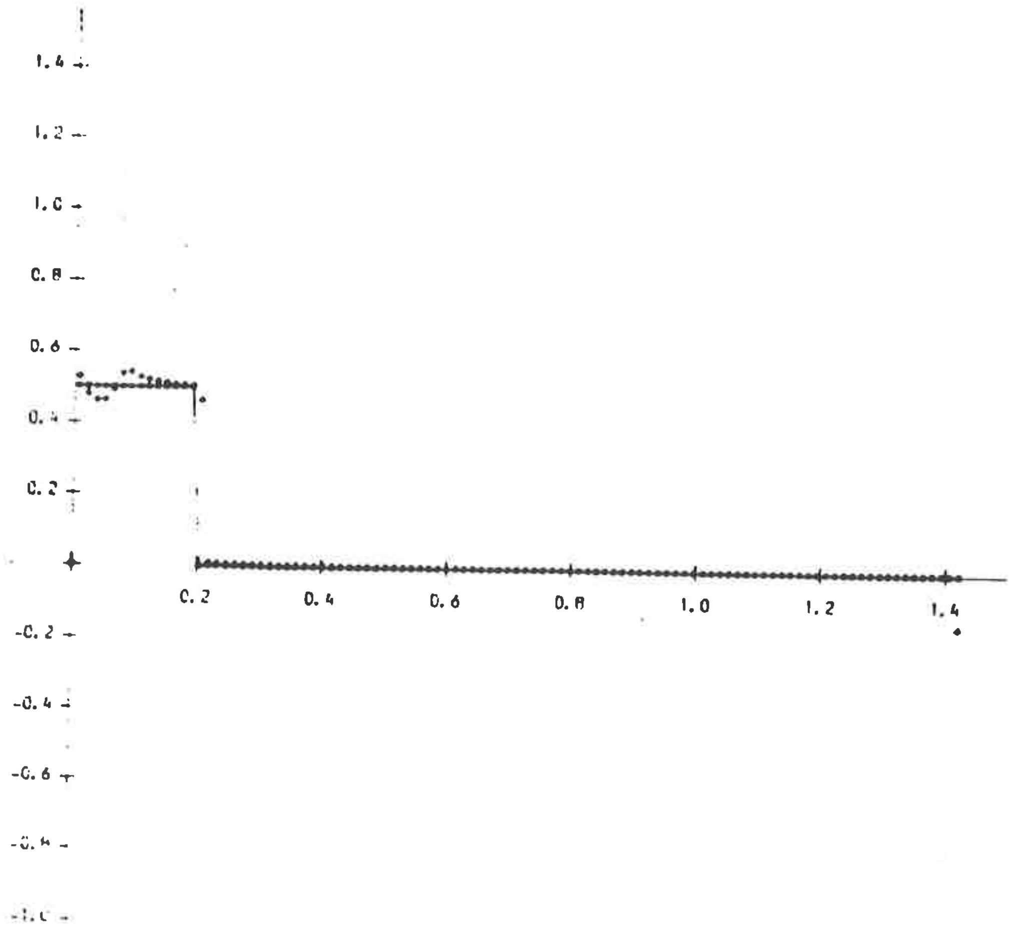


SOLUTION OF NCHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
ENERGY
AVERAGE ERROR : 0.00797
MAX ERROR : 0.55399
TIME : 0.6000
100 TIME STEPS

DT = 0.00600
DX = 0.01000
DY = 0.01000
MAX CFL : 0.39479
MAX HEIGHT : 0.55553
MIN HEIGHT : -0.18633

Figure 19

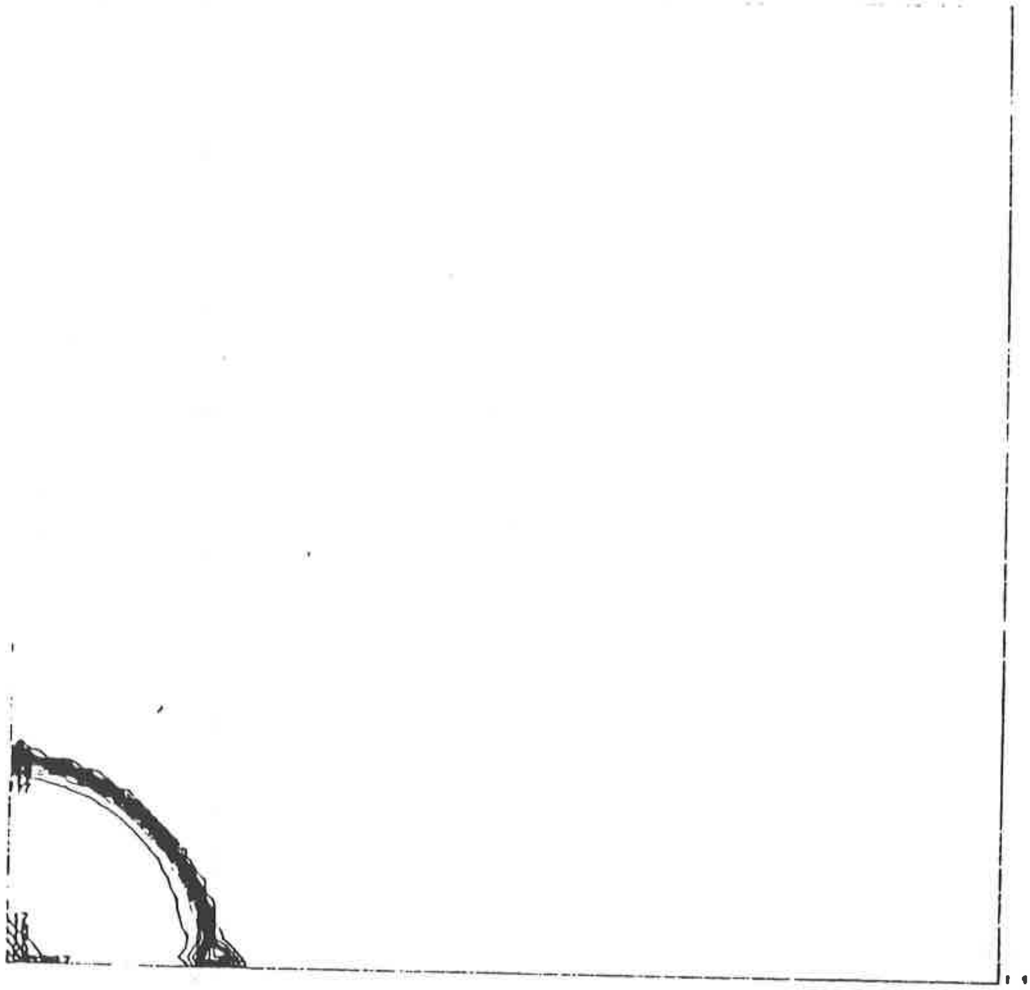


SOLUTION OF NCHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
ENERGY
AVERAGE ERROR : 0.00797
MAX ERROR : 0.55399
TIME : 0.6000
100 TIME STEPS

DT = 0.00600
DX = 0.01000
DY = 0.01000
MAX CFL : 0.39479
MAX HEIGHT : 0.55553
MIN HEIGHT : -0.18633

Figure 20

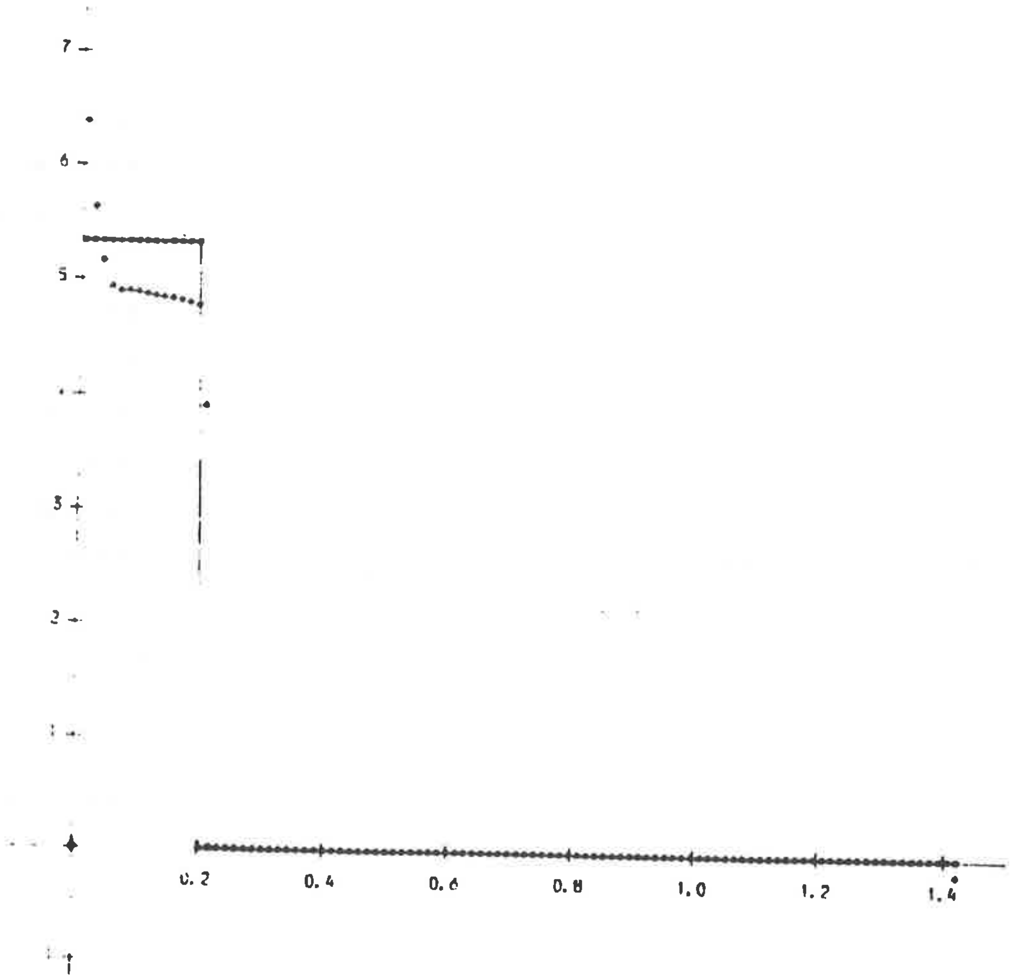


SOLUTION OF NOHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
PRESSURE
AVERAGE ERROR : 0.03632
MAX ERROR : 4.60337
TIME : 0.6000
100 TIME STEPS

DT = 0.00600
DX = 0.01000
DY = 0.01000
MAX CFL : 0.39479
MAX HEIGHT : 6.37933
MIN HEIGHT : -0.19801

Figure 21

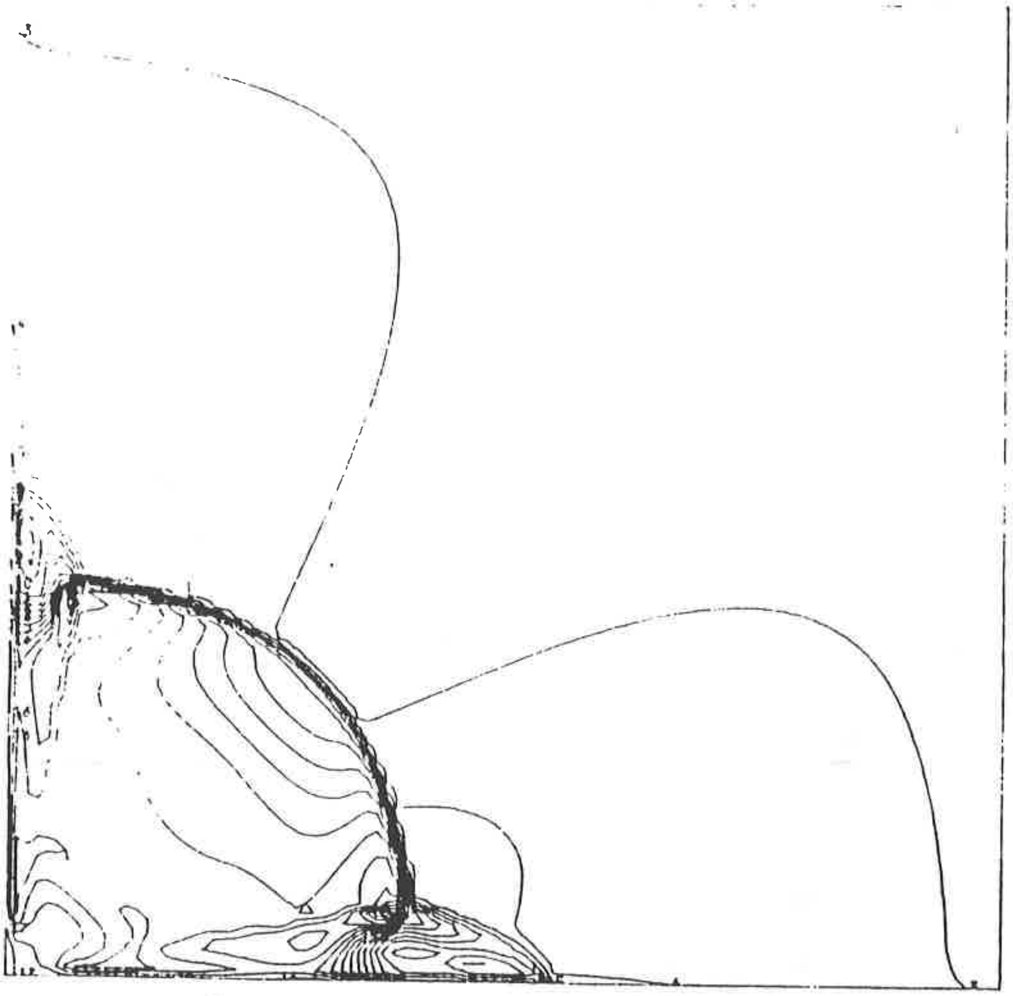


SOLUTION OF NOHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
PRESSURE
AVERAGE ERROR : 0.03632
MAX ERROR : 4.60337
TIME : 0.6000
100 TIME STEPS

DT = 0.00600
DX = 0.01000
DY = 0.01000
MAX CFL : 0.39479
MAX HEIGHT : 6.37933
MIN HEIGHT : -0.19801

Figure 22

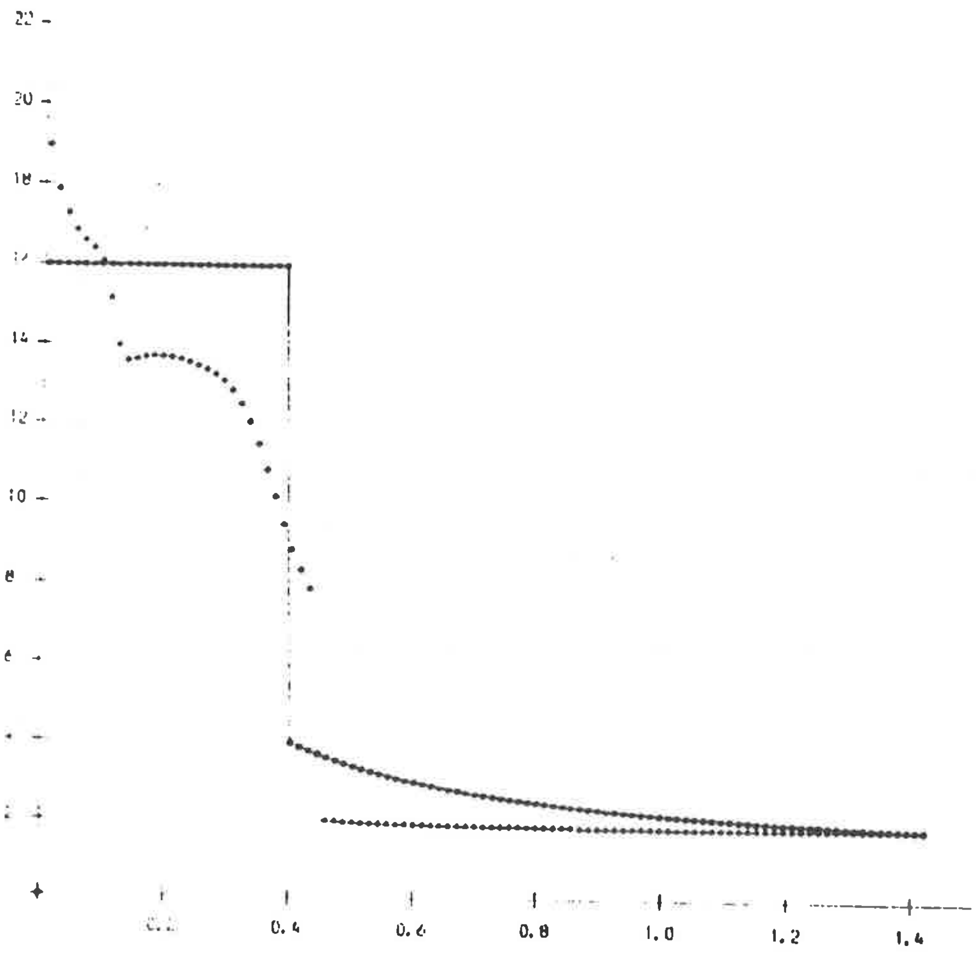


SOLUTION OF NUHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY
AVERAGE ERROR . 0.64275
MAX ERROR . 17.80502
TIME . 1.2000
200 TIME STEPS

DT = 0.00600
DX = 0.01000
DY = 0.01000
MAX CFL . 0.49810
MAX HEIGHT . 22.59927
MIN HEIGHT . 0.00000

Figure 23

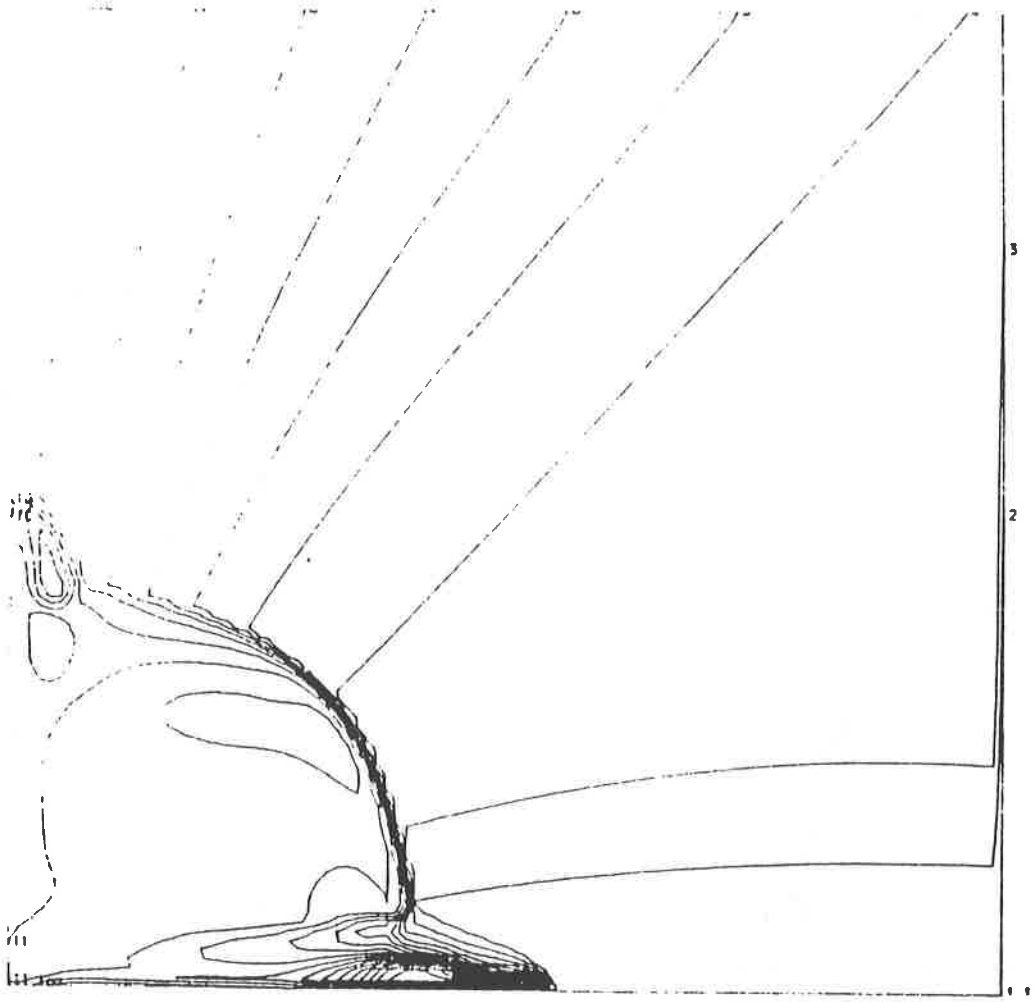


SOLUTION OF NUHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
DENSITY
AVERAGE ERROR . 0.64275
MAX ERROR . 17.80502
TIME . 1.2000
200 TIME STEPS

DT = 0.00600
DX = 0.01000
DY = 0.01000
MAX CFL . 0.49810
MAX HEIGHT . 22.59927
MIN HEIGHT . 0.00000

Figure 24



SOLUTION OF NOHS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

X-VELOCITY

AVERAGE ERROR = 0.17448

MAX ERROR = 1.74768

TIME = 1.2000

200 TIME STEPS

DT = 0.00600

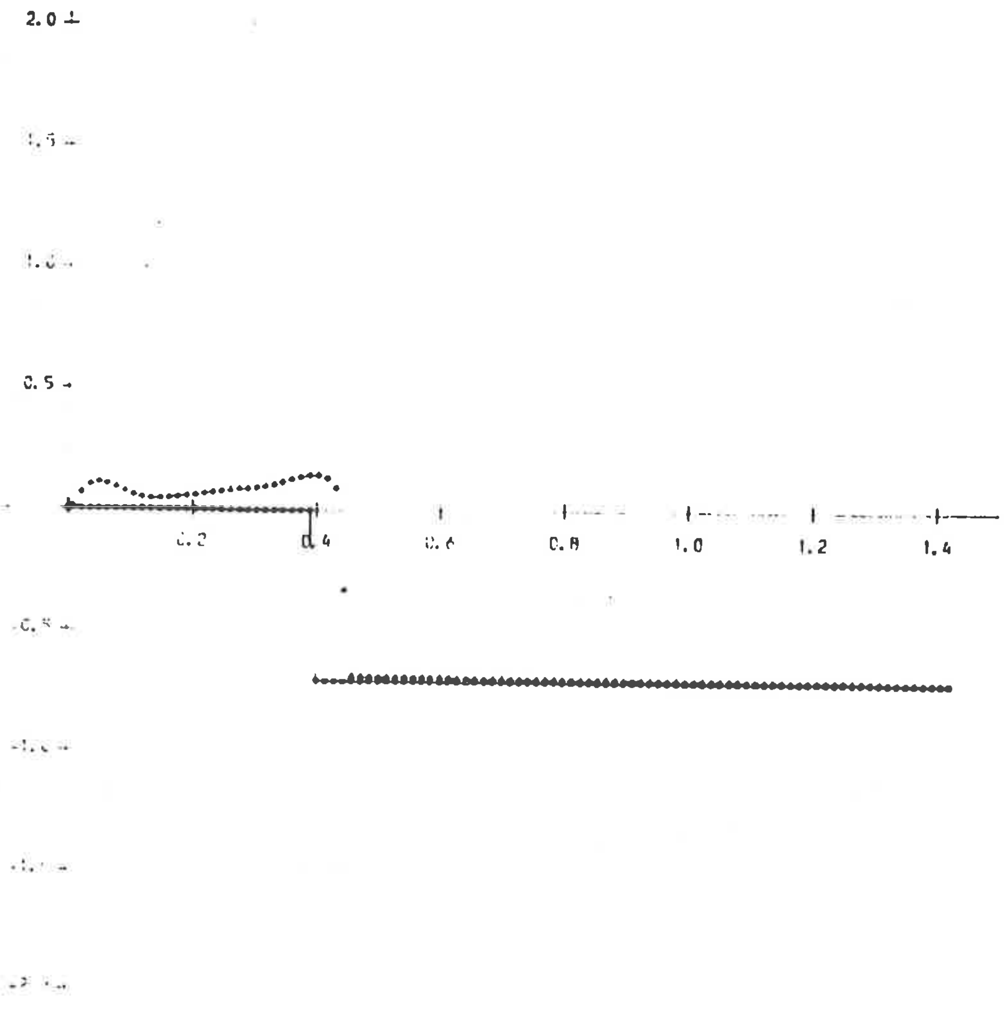
DX = 0.01000

DY = 0.01000

MAX CFL = 0.49810

MAX HEIGHT = 1.04057

MIN HEIGHT = -0.99999



SOLUTION OF NOHS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

X-VELOCITY

AVERAGE ERROR = 0.17448

MAX ERROR = 1.74768

TIME = 1.2000

200 TIME STEPS

DT = 0.00600

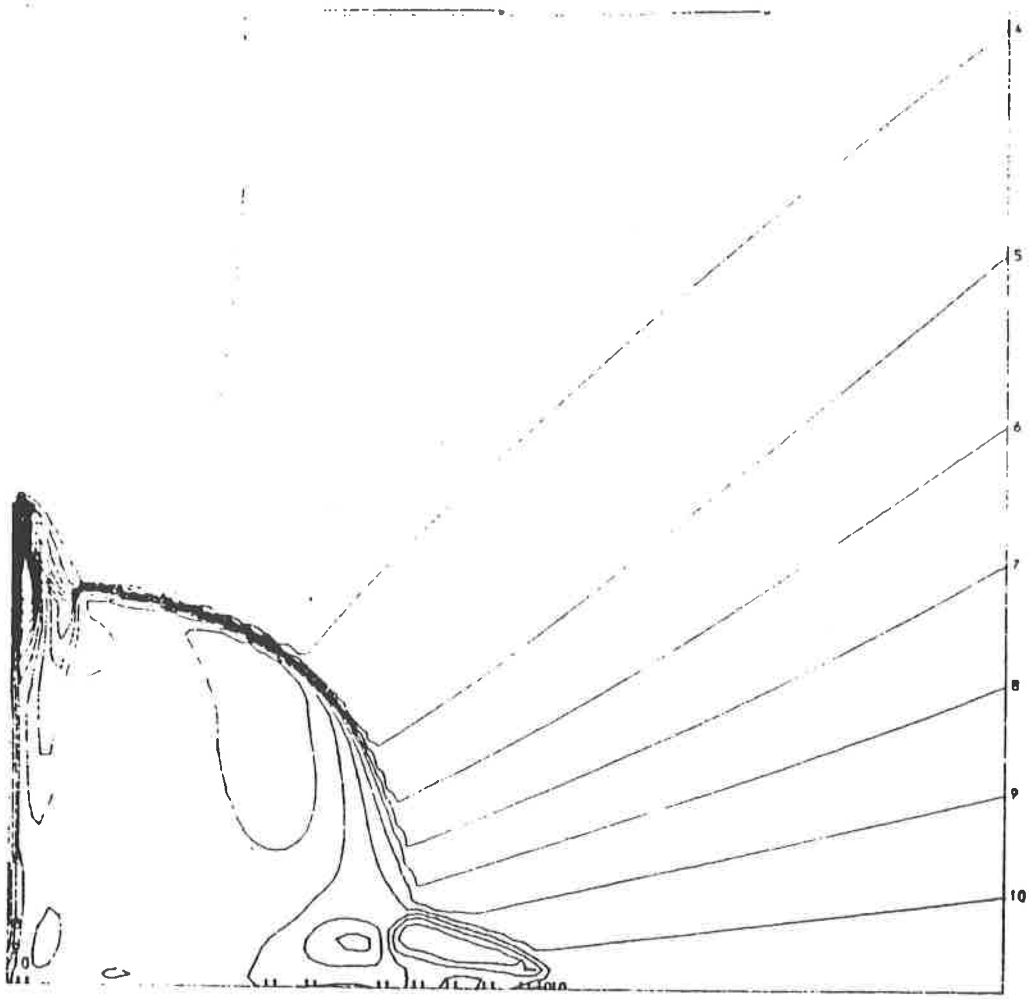
DX = 0.01000

DY = 0.01000

MAX CFL = 0.49810

MAX HEIGHT = 1.04057

MIN HEIGHT = -0.99999

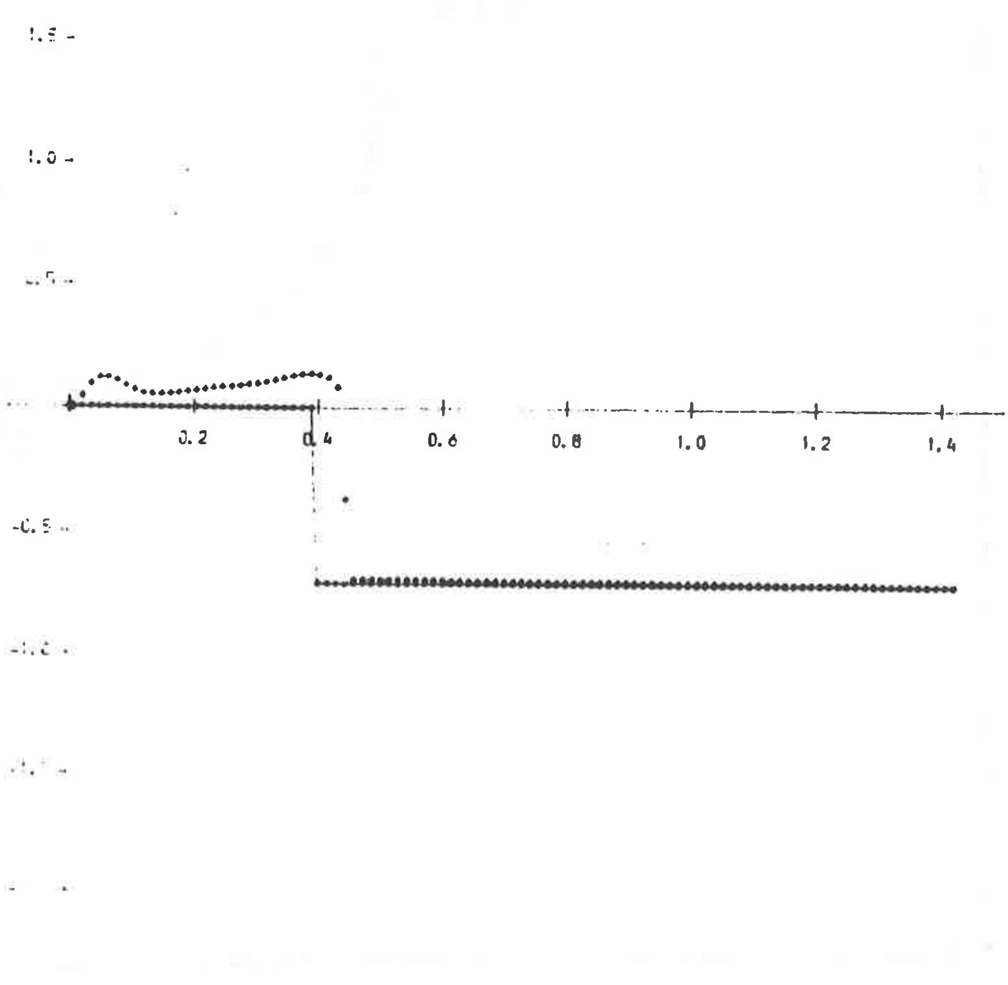


SOLUTION OF NCHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
Y-VELOCITY
AVERAGE ERROR : 0.17371
MAX ERROR : 1.70431
TIME : 1.2000
200 TIME STEPS

DT = 0.00600
DX = 0.01000
DY = 0.01000
MAX CFL : 0.49810
MAX HEIGHT : 0.99720
MIN HEIGHT : -0.99999

Figure 27

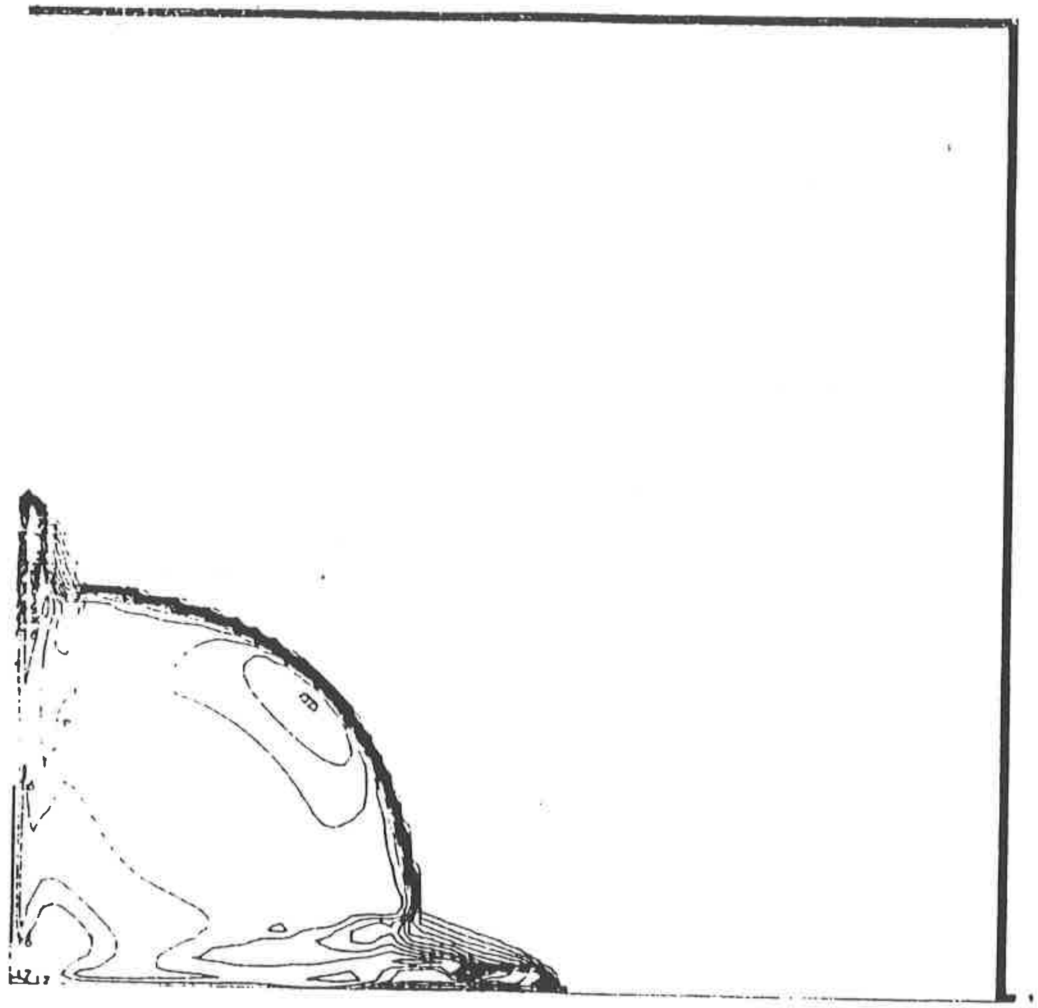


SOLUTION OF NCHS PROBLEM
USING (2) OPERATOR
SPLITTING AND FIRST ORDER
UPWIND DIFFERENCING.

OUTPUT FOR :-
Y-VELOCITY
AVERAGE ERROR : 0.17371
MAX ERROR : 1.70431
TIME : 1.2000
200 TIME STEPS

DT = 0.00600
DX = 0.01000
DY = 0.01000
MAX CFL : 0.49810
MAX HEIGHT : 0.99720
MIN HEIGHT : -0.99999

Figure 28



SOLUTION OF NOHS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

ENERGY

AVERAGE ERROR . 0.02246

MAX ERROR . 0.63380

TIME . 1.2000

200 TIME STEPS

DT = 0.00600

DX = 0.01000

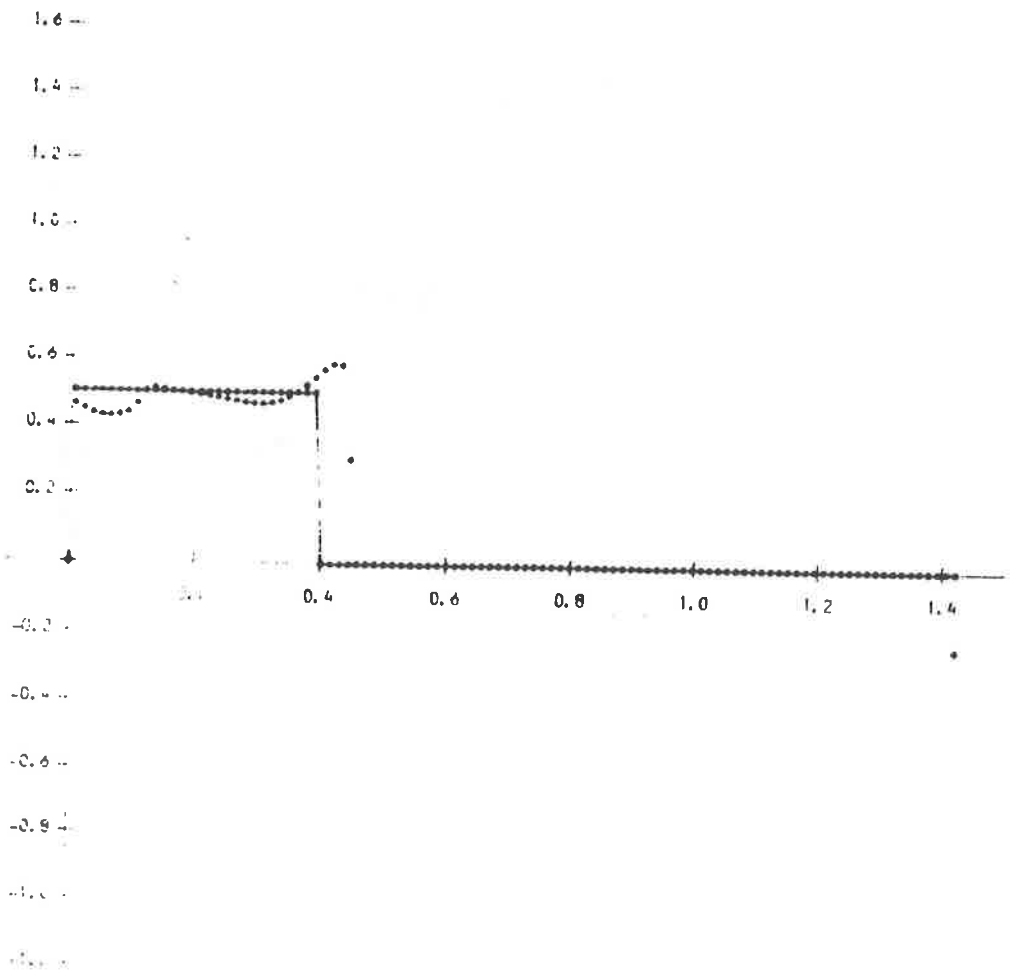
DY = 0.01000

MAX CFL . 0.49810

MAX HEIGHT . 0.63380

MIN HEIGHT . -0.27180

Figure 29



SOLUTION OF NOHS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

ENERGY

AVERAGE ERROR . 0.02246

MAX ERROR . 0.63380

TIME . 1.2000

200 TIME STEPS

DT = 0.00600

DX = 0.01000

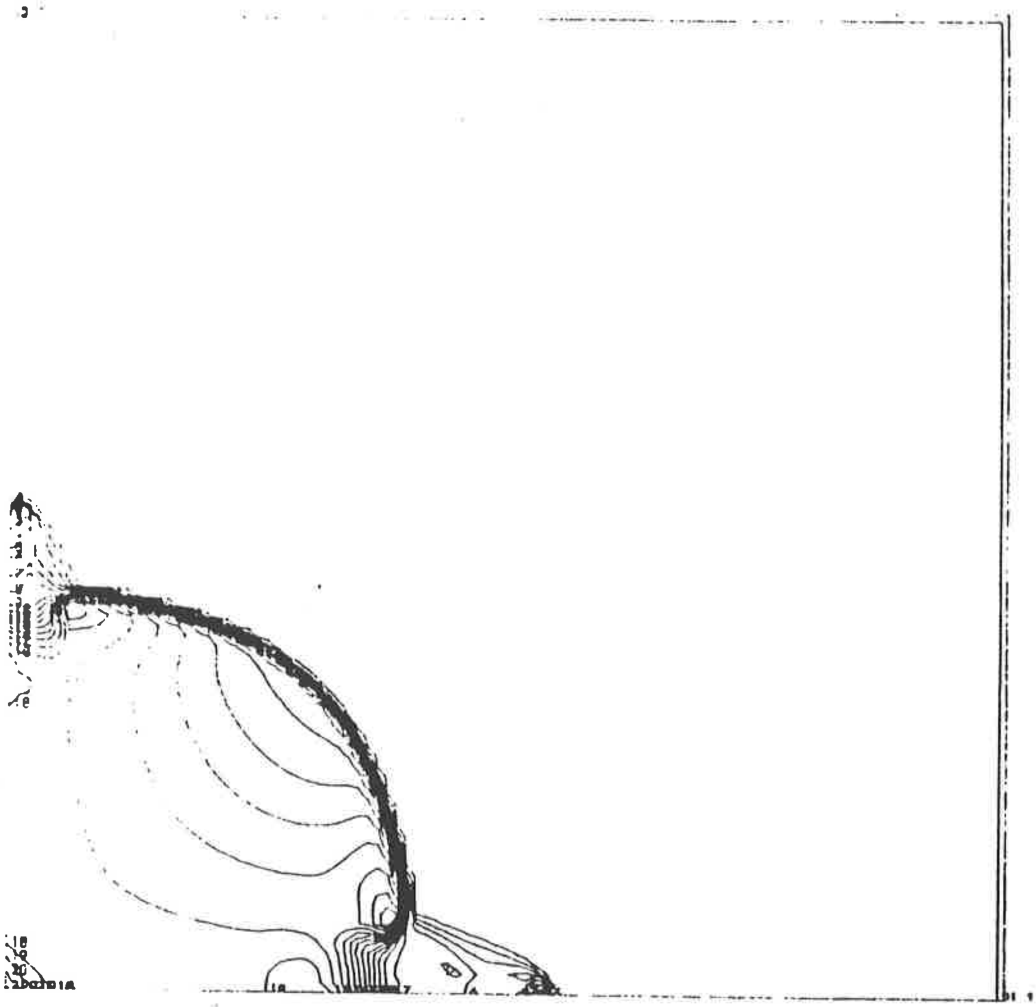
DY = 0.01000

MAX CFL . 0.49810

MAX HEIGHT . 0.63380

MIN HEIGHT . -0.27180

Figure 30



SOLUTION OF NOHS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

PRESSURE

AVERAGE ERROR : 0.20165

MAX ERROR : 5.38336

TIME : 1.2000

200 TIME STEPS

DT = 0.00600

DX = 0.01000

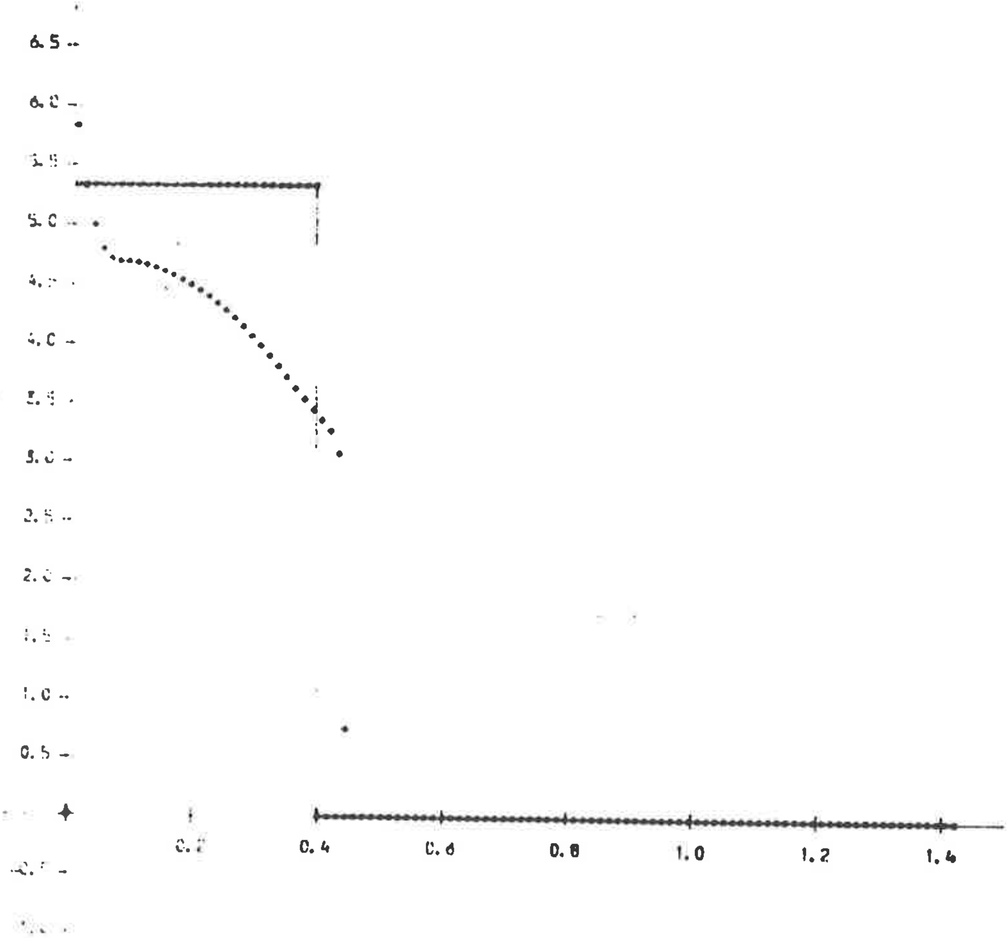
DY = 0.01000

MAX CFL : 0.49810

MAX HEIGHT : 5.83020

MIN HEIGHT : -0.39701

Figure 31



SOLUTION OF NOHS PROBLEM

USING (2) OPERATOR

SPLITTING AND FIRST ORDER

UPWIND DIFFERENCING.

OUTPUT FOR :-

PRESSURE

AVERAGE ERROR : 0.20165

MAX ERROR : 5.38336

TIME : 1.2000

200 TIME STEPS

DT = 0.00600

DX = 0.01000

DY = 0.01000

MAX CFL : 0.49810

MAX HEIGHT : 5.83020

MIN HEIGHT : -0.39701

Figure 32

Acknowledgements

I would like to thank Dr. M.J. Baines for many useful discussions. I would also like to thank Dr. P.K. Sweby and P. Glaister for useful interjections, and also R. Evans for a useful programming hint.

I acknowledge the financial support of S.E.R.C. and A.W.R.E. Aldermaston.

References

[1] W. F. Noh (1983)
Artificial Viscosity (Q) and Artificial Heat Flux (H) Errors for Spherically Divergent Shocks.
UCRL Preprint 89623.

[2] P. Glaister (1985)
Flux Difference Splitting Techniques for the Euler Equations in Non-Cartesian Geometry.
Numerical Analysis Report 8/85, University of Reading.

[3] P. Glaister (1986)
Similarity Solutions for Shock Reflection Problems in Gas Dynamics.
Numerical Analysis Report 13/86, University of Reading.

[4] P. L. Roe and J. Pike (1984)
Efficient Construction and Utilisation of Approximate Riemann Solutions.
Computing Methods in Applied Sciences and Engineering VI.
Ed. R. Glowinski and J. L. Lions, North Holland.

[5] P. Glaister (1986)
An approximate Linearised Riemann Solver for the Euler Equations in One Dimension with a General Equation of State.
Numerical Analysis Report 7/86, University of Reading.

[6] P. L. Roe (1981)
Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes.
J. Comput. Phys. 27, p357.

[7] N. N. Yanenko (1971)
The Method of Fractional Steps.
Springer-Verlag, New York.

[8] G. A. Sod (1985)
Numerical Methods in Fluid Dynamics. Initial and Initial Boundary Value Problems.
Cambridge University Press.

[9] G. A. Sod (1978)
A Survey of Several Finite Difference Methods for Systems of Non-Linear Hyperbolic Conservation Laws.
J. Comput. Phys. 27 p 1.

[10] J. Von Neumann and R. D. Richtmeyer (1950)
J. App. Phys 21. p232.

[11] W. D. Schulz (1964)
J. Phys. Vol. 5 no.1 p.p 133-138.

[12] P. Collella and P. Woodward (1982)
The Piecewise Parabolic Method (PPM).
Lawrence Berkeley Laboratory, Berkely, CA, LBL-14661.

[13] P. K. Sweby (1984)
High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws.
SIAM Journal on Numerical Analysis Vol.21 no 5, p995.

[14] G. Strang (1968)

On the Construction and Comparison of Difference Schemes.

SIAM Journal on Numerical Analysis Vol. 5. no 3, p 506.

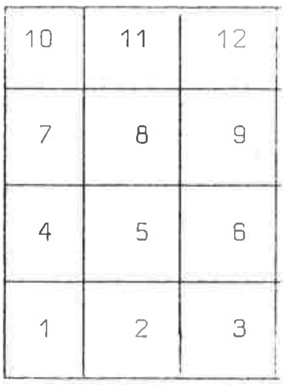
[15] P. L. Roe

Private Communication.

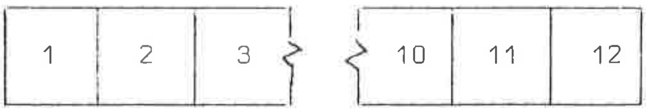
Appendix

A Note on Programming

Usually, when applying splitting to two-dimensional problems the method is to perform X-sweeps on the entire mesh, followed by Y-sweeps on the entire mesh. However, this tends to be inefficient, especially on large meshes. The inefficiency is due to the method employed by a computer to store arrays. A two-dimensional array is stored by a computer as a single, larger, one-dimensional array (see figure 1).



A "two"-dimensional array as 'seen' by programmer.



A "two"-dimensional array as stored by compter.

figure 1

During the actual development of the program I decided to store the 4 conservative variables at each mesh point in one three-dimensional array. So whilst the computer processed the mesh to update the solution, most of the array would be in virtual memory (ie on disc) and, as can be seen from figure 1, performing the Y-sweeps tended to be very inefficient because most of the elements that were required to update the solution were in virtual memory and thus a lot of time was wasted in waiting for the swopper.

To overcome this the following routine was used:-

Let X-sweep (I) be the 'subroutine' to perform updates $u^{n+1} = L_x(u^n)$ along the line whole $y = I\Delta y$ and let Y-sweep (I,J) be the 'subroutine' to perform a single update on the cell between $y = I\Delta y$ and $(I+1)\Delta y$ for $x = J\Delta x$. Then an efficient way of sweeping through the mesh is given by the following pseudo-program fragment:-

```
Perform  X-sweep (0)
For J = 1  to  JMAX-1        (JMAX x Δy = 1.0)
Perform  X-sweep (J)         (Lx^½)
For  I = 0  to  IMAX-1       (IMAX x Δx = 1.0)
Perform  Y-sweep (J,I)       (Ly)
NEXT I
Perform  X-sweep(J-1)        (Lx^½)
NEXT J
```

This can be suitably adjusted to encompass second order operator split schemes.