AN IMPLICIT REGRIDDING ALGORITHM WITH AN UNDERLYING FIXED GRID, FOR FRONT, SHOCK OR BOUNDARY TRACKING.

By D. E. Heath

Numerical Analysis Report 16 /87

University of Reading

Department of Mathematics

P O BOX 220

READING RG6 2AX

The work in this report forms part of the research program of the Institute for Computational Fluid Dynamics.

Contents

Abstract

In this report we develop two similar numerical algorithms for the solution of partial differential equations in which it is important to track a moving boundary, a shock or a steep front. These methods are based on a combination of a fixed grid and a local moving grid, each having the property that they leave the mesh unaltered once the front has passed through the region.

1. Introduction

In recent years much effort has been put into adaptive mesh algorithms for partial differential equations whose efficient solution requires good resolution in particular parts of the region where important features occur. Certain adaptive mesh methods, however, used for the numerical solution of moving boundary problems or hyberbolic problems which develop shocks have the unfortunate property that the solution region is intrinsically split into two domains, the domains being separated by the shock or the moving boundary (e.g. the Moving Finite Element Method (MFE) (see [1] & [2])) The main disadvantage with this property is that the nodes are forced to lie in one of two separated regions, with no nodal transfer allowed between them. Although some very good results have been achieved using this technique (see e.g. Moody [3])), certain requirements have to be met before the method will work (e.g. the moving boundary or shock must lie in the solution domain at the start of the problem.).

For certain types of problem the moving boundary does not form until after a certain time, or if it does exist it is just inside the solution domain: both of these conditions will cause problems if attempted to be solved by the above mentioned techniques. The first case will be impossible to solve if we consider two regions, as one of the regions will contain no elements, the second condition (i.e. the boundary is initial just inside the solution region) could cause problems, if for example the physical geometry played an important role

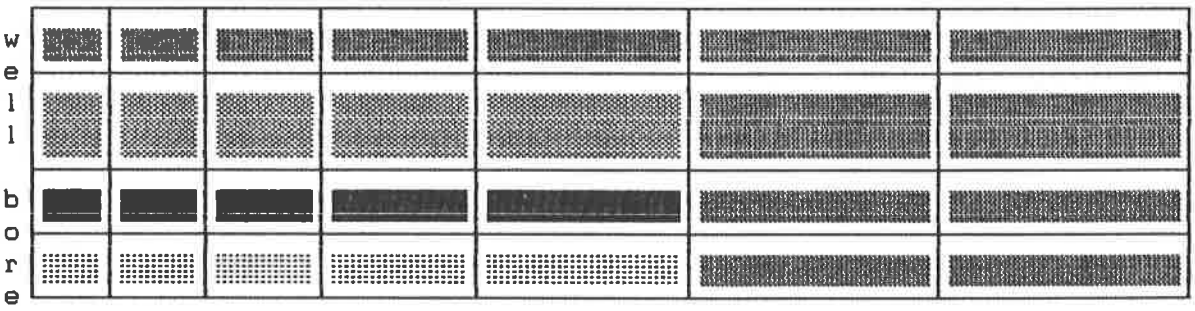
at the boundary of the region. For example in oil reservoir modelling, if we considered the injection of steam from a well bore into a reservoir, then as well as trying to track the moving interface between the steam and the oil/water through the reservoir, it is also important to have a large number of elements present around the well bore at all times as vast changes in pressure occur around this point throughout the injection and production phases. If two regions were employed, one either side of the interface, than as the boundary moved into the reservoir their would be insufficient elements left around the well bore, in order to represent the solution.

In section 2 of this report we describe the MFE method and the manner in which the nodal velocities are constrained. Section 3 develops two numerical methods capable of transferring nodes from one region to another, for one-dimensional moving boundary problems. The only moving nodes present in this method are in the group which we use to track the boundary. After the interface passes through the region the nodes return to their initial positions. The main reason for not allowing all the nodes to move in each region (as Moody does in [3]) is again related to the physical problem. Since it is intended to use this method for oil reservoir modelling problems (e.g. steam injection) and since the rock properties are inhomogeneous, it is desirable to keep the nodes in set domains (see Fig 1.1).

In section 4 we apply the methods of sections 2 & 3 to some test problems, firstly to a moving boundary problem, secondly to the case of tracking a steep front in the solution of the viscous Burgers' equation, and finally to shock problems

(namely linear advection and inviscid Burgers') where the theory of section 3 has to be extended so as to cope with double valued nodes. Finally we consider the Buckley-Leverett equation, representing the jump in saturation either as a shock and or as a steep front. The conclusions about the two methods are contained in section 5.

Fig 1.1 Diagram showing how rock structure can affect the grid.



Key

- Shale.
- Sandstone.
- Limestone.
- Fine sand.

2. Front tracking

It is important in many problems (e.g. those involving moving boundaries) to be able to locate accurately a certain point in the solution domain, e.g. the position of the boundary. Numerically, this is best achieved by use of an adaptive mesh method, which allows nodal positions to vary during the solution, for example the Moving Finite Element method (MFE), (see [1] [2] & [4]). In this kind of method, at the end of each time step a node may be located at the moving boundary, the shock etc.

2.1 MFE with constrained nodal velocities

In this section we will show how nodal velocities can be incorporated into the standard Galerkin equations, where the speeds of the nodes are specified.

Consider the standard evolutionary equation

$$u_t - L(u) = 0 \qquad (2.1)$$

where L is some spatial differential operator (which may be non-linear). We seek a semi-discrete approximation v to the solution u of (2.1) of the form

$$v = \sum_{i=0}^{i=N} v_i \alpha_i \qquad (2.2)$$

where v_i are coefficients (nodal amplitudes) of the

corresponding nodes $s_i = s_i(t)$, with $\underline{s} = (s_0, s_1, \ldots, s_N)$, and $\alpha_i = \alpha_i(x, \underline{s}(t))$ are the standard piecewise linear basis functions, on the grid defined by s_i. A typical basis function ($0 < i < N$) can be seen in fig.2.1.

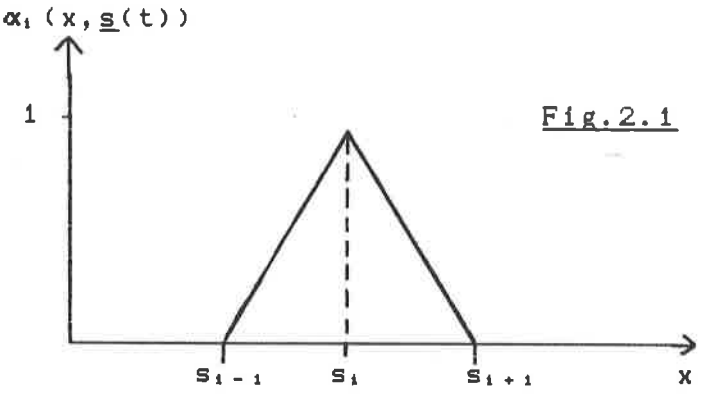


Fig.2.1

Here,

$$\alpha_i = \begin{cases} \dfrac{x - s_{i-1}}{s_i - s_{i-1}} & s_{i-1} \leq x < s_i \\ \dfrac{s_{i+1} - x}{s_{i+1} - s_i} & s_i < x \leq s_{i+1} \\ 0 & \text{otherwise.} \end{cases} \qquad (2.3)$$

Differentiating (2.1) with respect to time,

$$v_t = \sum_{i=0}^{i=N} (\dot{v}_i \alpha_i + \dot{s}_i \beta_i) \qquad (2.4)$$

where $\beta_i = \beta_i(x, \underline{s}(t), \underline{v}(t))$ are piecewise linear discontinuous basis functions (see Miller & Miller [5] or Wathen & Baines [1]). For the given α_i it has been shown by Lynch [6] that

$$\beta_i = -v_x \alpha_i \ : \qquad (2.5)$$

using the notation

$$m_i = \frac{v_{i+1} - v_i}{s_{i+1} - s_i} \qquad (2.6)$$

we have

$$\beta_i = \begin{cases} -m_{i-1}\alpha_i & s_{i-1} \le x < s_i \\ -m_i\alpha_i & s_i < x \le s_{i+1} \\ 0 & \text{otherwise.} \end{cases} \qquad (2.7)$$

A typical β_i $(0<i<N)$ basis function can be seen in Fig.2.2

$\beta_i(x, \underline{s}(t), \underline{v}(t))$

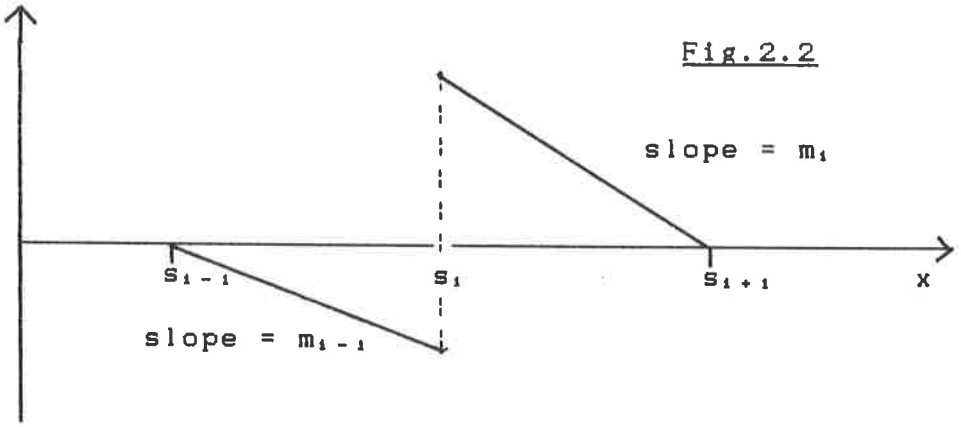


Fig.2.2

Equations determining the N+1 unknowns (v_i) are obtained by minimisation of the square of the L_2 norm of the residual, namely

$$\| v_t - L(v) \|_2^2 , \qquad (2.8)$$

with respect to the $\dot{v}_i$ $(i=0,\ldots,N)$, taking $\dot{s}_i$ $(i=0,\ldots,N)$ as parameters (determined by the moving boundary, shock, etc.(see section 4)), which gives rise to the Galerkin equations

$$< v_t - L(v) , \alpha_i > = 0 \qquad i = 0,\ldots,N \qquad (2.9)$$

where

$$< \cdot , \cdot > = \int_{\text{domain}}^{\text{physical}} \cdot \, . \, \cdot \, d\tau \qquad (2.10)$$

Substitution of (2.4) into (2.9) gives the set of N+1 semi-discrete equations

$$< \sum_{j=0}^{j=N} (\dot{v}_j \alpha_j + \dot{s}_j \beta_j) + L(v) , \alpha_i > = 0 \quad i=0,\ldots,N \qquad (2.11)$$

$$\sum_{j=0}^{j=N} \{ \dot{v}_j < \alpha_j , \alpha_i > + \dot{s}_j < \beta_j , \alpha_i > \} + < L(v) , \alpha_i > = 0$$

$$i=0,\ldots,N \qquad (2.12)$$

Assuming L(v) is a linear operator, by use of an Euler time-stepping scheme (using a θ implicit-explicit factor) and using (2.3) & (2.7), the system of equations (2.12) may be written as

$$A\underline{x} = \underline{b} \qquad (2.13)$$

where A is some tridiagonal matrix (depending on θ), $\underline{x}$ is the solution vector $\underline{v}$ at the new time level and $\underline{b}$ is the right hand side vector again depending on θ & L(v). This system may be solved by use of a simple LU solver.

2.2 Evaluation of inner products

In this report we will consider general spatial operators L(u), including

$$L(u) = u_{xx} \tag{2.14}$$

As we are considering a piecewise linear approximation, the inner product of L(v) with α_i will have the form of integrals of delta functions. The technique we use to handle these inner products is the same as in the approach of Mueller [7] , which is well described in Moody [3]: therefore we only outline the method below. Evaluation of

$$< L(v) , \alpha_i > = < v_{xx} , \alpha_i > \quad (0<i<N) \tag{2.15}$$

using integration by parts gives

$$= [v_x \alpha_i]_{s_{i-1}}^{s_{i+1}} - \int_{s_{i-1}}^{s_{i+1}} v_x \alpha_{i,x} \, dx \tag{2.16}$$

where

$$\alpha_{i,x} = \begin{cases} \dfrac{1}{s_i - s_{i-1}} & s_{i-1} \leq x < s_i \\ \dfrac{-1}{s_{i+1} - s_i} & s_{i+1} < x \leq s_i \end{cases} \tag{2.17}$$

We then have

$$< v_{xx} , \alpha_i > = m_i - m_{i-1} \tag{2.18}$$

as $[v_x \alpha_i]_{s_{i-1}}^{s_{i+1}} = 0$, owing to the basis functions α_i being zero at the two limits.

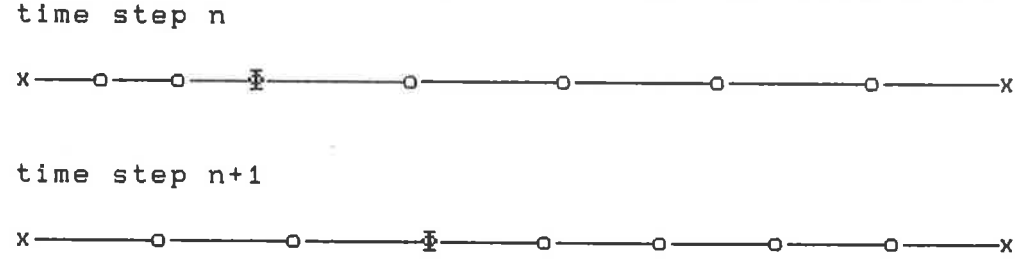
2.3 Nodal velocities

The MFE method has the nodal velocities $\dot{s}_i$ as unknowns, the v_i and s_i being solved for by minimising (2.3) over the amplitudes $\dot{v}_i$ and the velocities $\dot{s}_i$ (i=0,...,N) to give a double system of Galerkin equations. However, the type of problems we are interested in here are those in which we are trying to track a moving boundary, shock or step front, the speed of which can usually be obtained or estimated from the physical equations. Therefore the velocities of certain nodes can be determined in advance such that they follow the boundary, shock, etc.

The main constraint with the MFE type of approach is node overtaking. Boundary conditions usually require a fixed node at either end of the solution domain, and with all interior nodes moving at the boundary velocity, nodes will soon crash into the fixed node. Moody [3] overcomes this problem by allowing only the node at a moving boundary to move with the boundary velocity, the remaining nodes being constrained such that their velocities are linked to this boundary velocity. Thus at the end of each time-step the nodes are equally spaced in either domain each side of the boundary. (see fig.2.3)

Fig.2.3.

Key.
x - Fixed node.
o - Moving node.
Φ - Boundary node.



This method gives good results but is limited in scope as follows. If we consider the nodal distribution of Fig.2.3. at a later time, as in Fig.2.4.

Fig.2.4.

we see that the nodal spacing is now very small at one end and large at the other, which causes problems both in numerical stability and physical representation. In the next section of this report we will outline two regridding algorithms that allow a moving mesh to be superimposed on a fixed grid, thus allowing tracking of the boundary, shock etc. by use of the methods outlined above, as well as resolving physical representation and numerical stability by use of the fixed grid.

3. Regridding Techniques

In this section we describe two different regridding algorithms that allow a moving and a fixed grid to be combined into a single grid. Explicit Euler time-stepping will be used throughout this section, although a significant implicit version is described in section 4.

Let F_0 denote the initial fixed grid, whose nodal distribution is based on the initial data, the physical geometry and any numerically restrictions (i.e. element size). (See Fig.3.1, using the notation of section 2.3). Now let F_t be a sub-grid of F_0 at time t, containing all but a few of the nodes of F_0, (initially $F_t \equiv F_0$). The moving grid denoted by M_t consists of a group of elements (2 or 3 for a moving boundary problem, but perhaps up to 10 for tracking a steep front (see section 4)) which follow the moving boundary, front, etc. through the region (see Fig.3.2.). The grid G_t is formed by the superposition of M_t on F_t (see Fig.3.3.), being the grid used at time t for the solution of the Galerkin equations of section 2.1. The velocity of the nodes on grid F_t

Fig.3.1. Fixed grid F_0

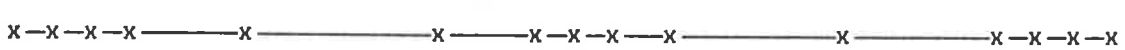
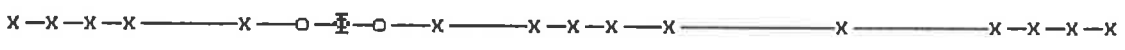


Fig.3.2. Moving grid M_t

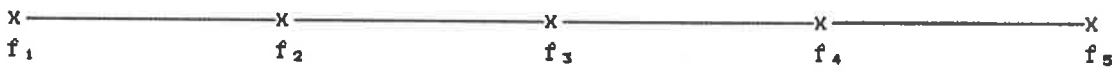
Fig.3.3. Combined grid G_t



are zero and the velocity of the nodes on grid M_t are calculated such that nodes follow the moving boundary, front, etc.

We first consider a simple fixed grid F_0 (see Fig.3.4.) with nodal positions f_i , i=1,...,5 say, and initially $F_t \equiv F_0$,

Fig.3.4. Grid F_0



and an initial moving grid M_0 (see Fig.3.5.) with nodal positions m_i , i=1,2,3 say.

Fig.3.5. Grid M_0 nodal motion

Then define G_t as the solution grid at time t, where the nodes on grid F_t ($F_t \equiv F_0$ for all t) are fixed and the speed of the nodes on grid M_t are determined by the velocity of the moving boundary at time t. The solution progresses smoothly until, as in the above grid example, node m_3 gets too close to, or even overtakes, node f_3. If node m_3 actually overtakes node f_3 then the solution could be double valued at certain points, when only a singled value solution should exist (see Fig.3.6.).

The idea of the methods outlined in the next two subsections is to use regridding techniques which can combine the two grids F_t and M_t in such a way that nodes on the

moving grid M_t can "pass" through those on the fixed grid F_t, thus avoiding the problems outlined above.

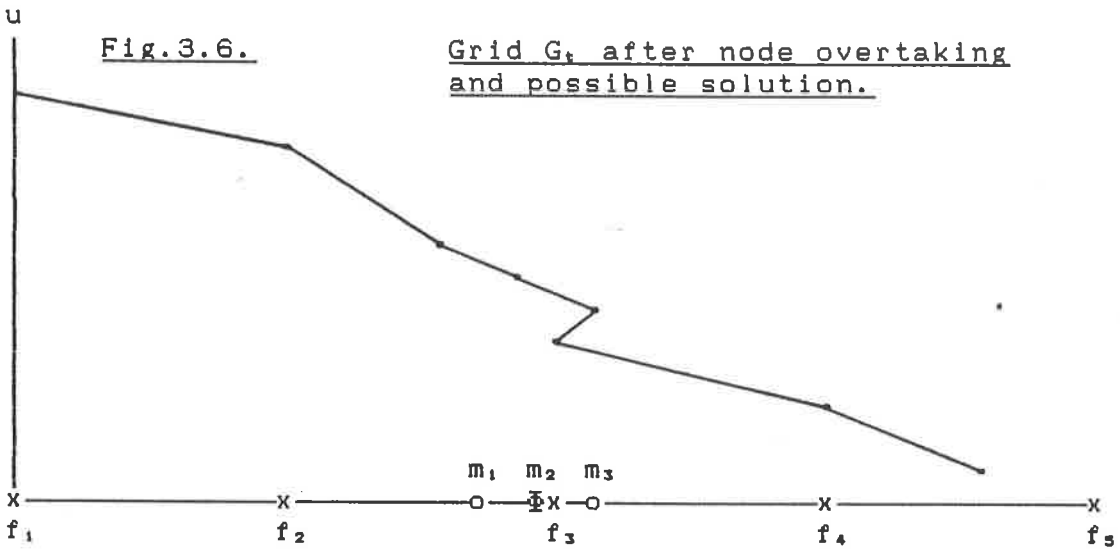


Fig.3.6. Grid G_t after node overtaking and possible solution.

3.1 Node Reversal

The first regridding technique we consider is a method called Node Reversal (NR). For simplicity we will just consider the operation of this method on a moving boundary problem although, as will be seen in section 4, it is also well suited to propagation of steep fronts and shocks (some special treatment is necessary for shock problems (see section 4.3))

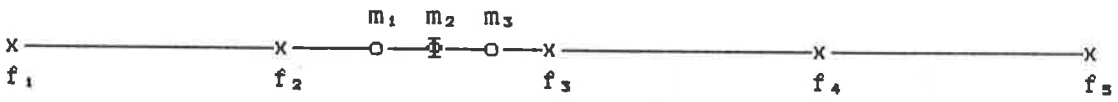
3.1.1 NR Algorithm

First we introduce a minimum nodal spacing δ, this being used for the nodes on grid M_t, and we assume that initially on grid G_t that the distance between node m_i and the node before

it, f_2 say, is $< \mathcal{E}$. We also assume that the distance between node m_3 and the next node f_3 say $< \mathcal{E}$. We call this situation Mode 0. In this case we carry out the following procedure:

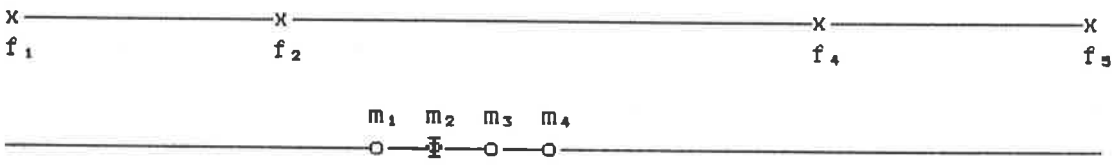
1) Using the boundary velocity for the speeds of the nodes on M_t advance the solution until the distance between m_3 & f_3 = $\mathcal{E}$. (see Fig.3.7.)

Fig.3.7. Grid G_t



2) Remove node f_3 from grid F_t and include it in grid M_t (see Fig.3.8)

Fig.3.8. Grid F_t & M_t after step 2



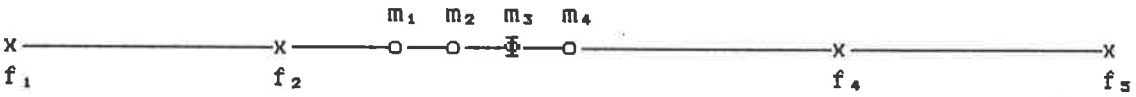
3) Using the equation for the boundary velocity, calculate the position of the moving boundary at the end of the next time-step (assuming explicit Euler time-stepping) s_b say. If $s_b > m_3$ adjust the time step dt such that $s_b = m_3$.

4) Calculate the speed for node m_3 to travel to s_b, $\dot{s}_n$ say ($\dot{s}_n = (s_b - m_3)/dt$).

5) Advance the solution one time-step with the speeds of the nodes on grid M_t set to $\dot{s}_n$ (N.B. $\dot{s}_n \leq 0$, i.e. the nodal direction is reversed).

6) The boundary position is now located at node m_3 (see Fig. 3.9.)

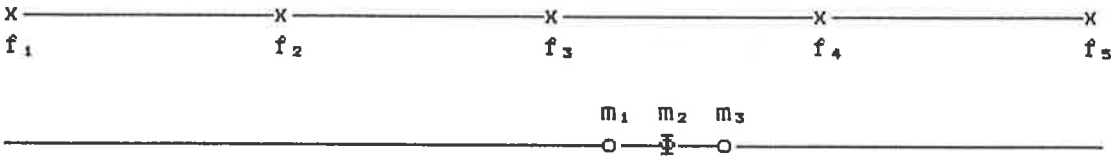
Fig.3.9. Grid G_t at step 6



7) Using the boundary velocity for the speeds of the nodes on M_t advance the solution until node m_1 is located at the position f_2 on the grid F_o. Call this Mode -1.

8) Remove node m_1 from grid M_t and place it back in grid F_t, and renumber the nodes in grid M_t (see Fig.3.10).

Fig.3.10. Grid F_t & M_t after step 8



9) Now grid $F_t \equiv F_o$ again, therefore in Mode 0 repeat from step 1.

3.2 Node Deletion

The second regridding algorithm we consider is called Nodal Deletion (ND). As in section 3.1 we will consider only a moving boundary problem, although this method adapts even more readily to shock problems than NR, as will be seen in section 4.3.

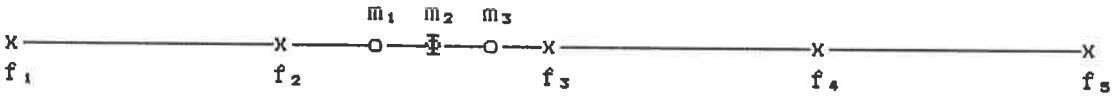
3.2.1 ND Algorithm

Although several steps of this algorithm are similar to those of the NR algorithm, for completeness we will include them all.

Spacing the nodes in grid M_t by $\mathcal{E}$, and as before, making sure initially that no node on grid M_t is within a distance $\mathcal{E}$ from the nodes on grid F_o, i.e. we are in Mode 0, we carry out the following steps:

1) Using the boundary velocity for the speeds of the nodes on M_t advance the solution until the distance between m_3 & f_3 = $\mathcal{E}$. (see Fig.3.11.)

Fig.3.11. Grid G_t

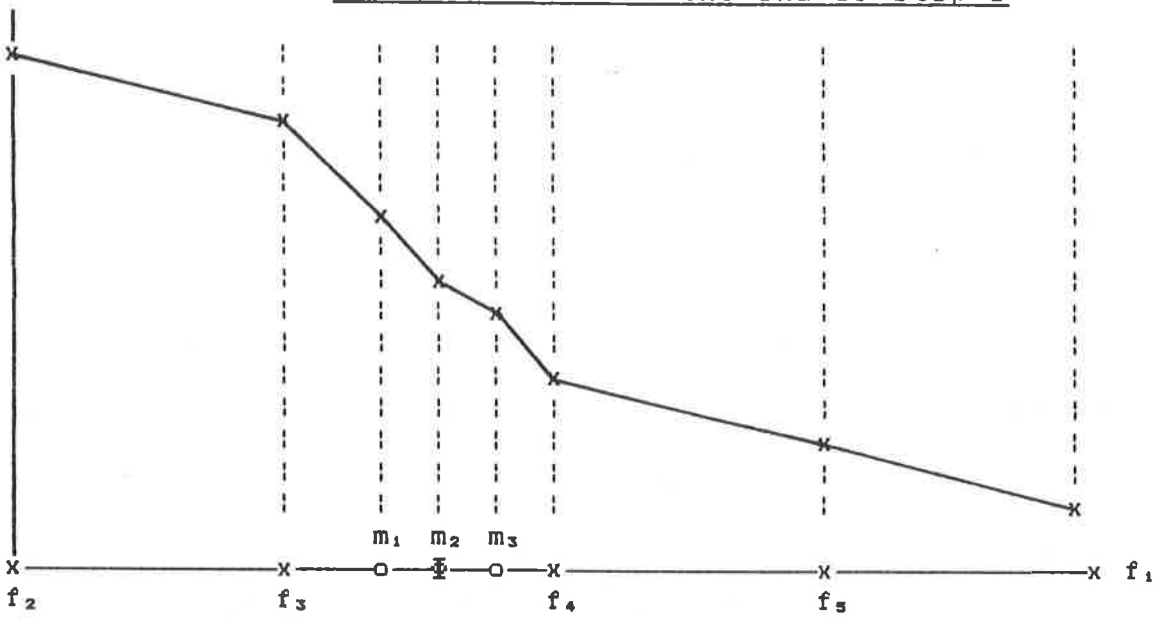


2) Delete node f_3 from grid F_t and adjust the nodal

amplitudes of nodes m_3 and f_4 so as to conserve the area under the curve and minimise the L_2 norm of the difference between the previous and present representations. (see Fig.3.12.)

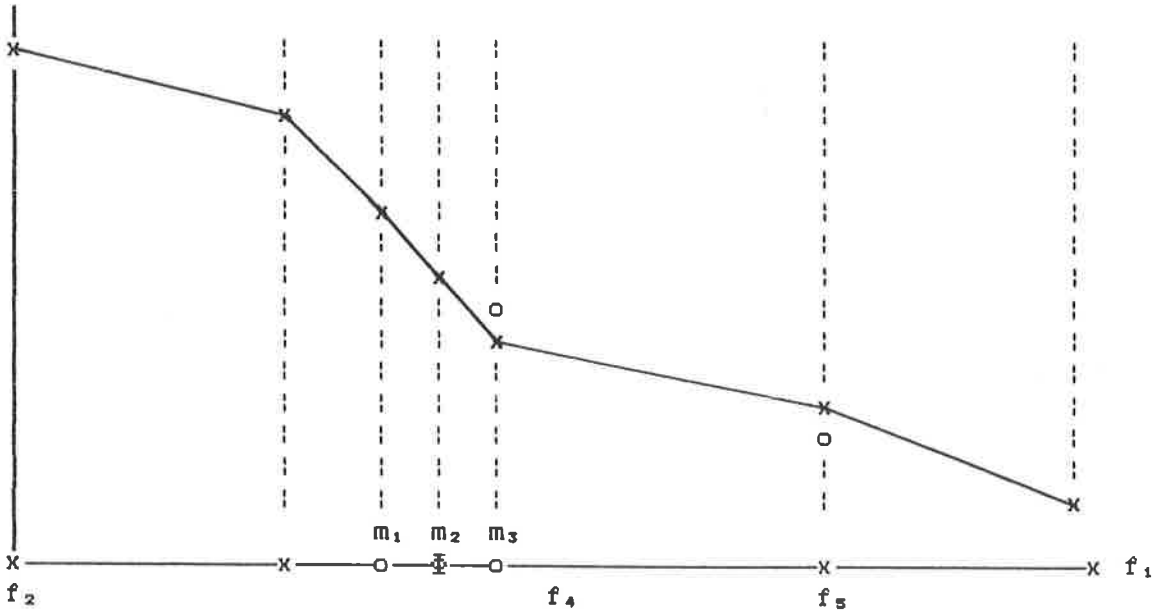
Fig.3.12

Representation at the end of step 1



Representation at the end of step 4.

o - representing the old amplitudes at nodes m_3 & f_4



3) Insert a node into grid M_t at a distance ξ behind node m_1 and renumber nodes such that the new node is m_1 (see Fig 3.13.).

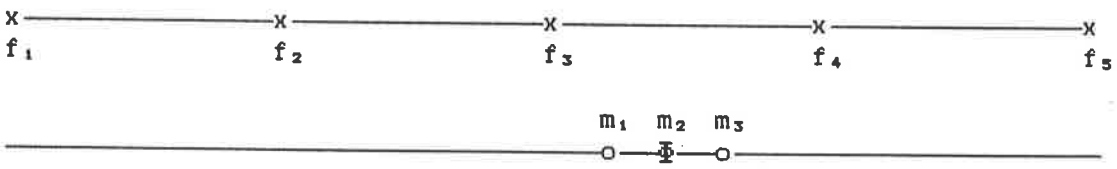
Fig.3.13. Grid M_t after step 3

4) Linearly interpolate the amplitude of node m_1 from nodes f_2 & m_2.

5) Using the boundary velocity for the speeds of the nodes on M_t advance the solution until node m_1 is located at the position f_2 on the grid F_o. (this is Mode -1.)

6) Remove node m_1 from grid M_t and place it back in grid F_t, and renumber the nodes in grid M_t. (see Fig.3.14)

Fig.3.14. Grid F_t & M_t after step 8



9) Grid $F_t \equiv F_o$ again and therefore in Mode 0, we may repeat from step 1.

4 Application of the NR and ND Algorithms to Test Problems and Special Procedures.

In this section of the report we apply the methods described in the previous sections to certain basic test problems, the first being a general moving boundary problem, both in cartesian and radial geometries, and the second being the tracking of a steep front. Then the solutions of shock problems are considered, first with simple linear advection and second with inviscid Burgers'. Finally the Buckley-Leverett equation is solved using the new techniques, both in a homogeneous form (a discontinuity having already formed) and with a viscous term (propagation of a steep front).

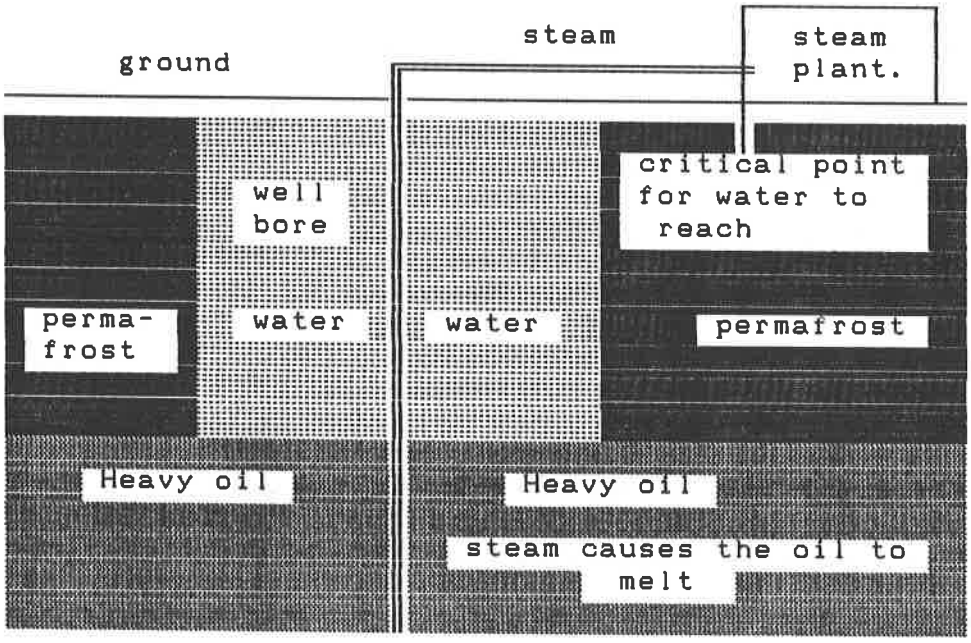
It was found necessary to enhance the two algorithms in order to obtain efficient solutions. In section 4.1.2 implicit versions of the two algorithms above are outlined with an implicit non-linear solver being discussed in section 4.2.2. Finally in order to be able to cope with double valued solutions (in shock modelling) the NR algorithm is modified, this being described in section 4.3.

4.1 Application to a Moving Boundary Problem.

The type of moving boundary problem we will consider here is a two-phase Stefan problem. This type of problem can be used to model several different physical process, the main one being the melting/freezing of water into ice, where the moving boundary represents the interface between water and ice. One application of this process occurs in permafrost thawing

around injection wells during heavy oil recovery (see [8]), where the high temperature in the well bore causes surrounding permafrost to melt. The accurate tracking of the ice/water interface is important to both the quality of steam entering the reservoir and the thawing of the permafrost at the surface, where important machinery may be located (see Fig.4.1).

Fig.4.1. Steam injection well through permafrost showing water/ice interface.



The model we consider is a two-phase Stefan problem in which forcing terms are present and is an amended version of that presented by Ciavaldini in [9] (see Moody [3]). It has been chosen because it contains a simple analytic solution which allows comparison with numerical results. Moreover the velocity of the moving boundary in this problem can be stated explicitly in terms of heat flux between the two phases. The problem may be stated as follows.

4.1.1 The Moving Boundary Problem

The equations governing the problem in the two regions are

$$\left.\begin{aligned} u_t &= k_L u_{xx} - s_0^2 e^t - 2k_L \qquad 0 < x < s(t) \\ u_t &= k_R u_{xx} - s_0^2 e^t - 2k_R \qquad s(t) < x < 1 \end{aligned}\right\} \; 0 < t < t_1 \qquad (4.1)$$

where $s(t)$ is the position of the moving boundary.

The fixed boundary conditions are

$$\left.\begin{aligned} u_x &= 0 \qquad x = 0 \\ u_x &= 2 \qquad x = 1 \end{aligned}\right\} \; 0 < t < t_1 \qquad (4.2)$$

and the boundary conditions at the moving boundary are

$$\left.\begin{aligned} u &= 0 \\ k_L u_x^- - k_R u_x^+ &= L\dot{s}(t) \end{aligned}\right\} \; 0 < t < t_1 \qquad (4.3)$$

with initial conditions.

$$u(x,0) = x^2 - s_0^2 \qquad 0 < x < 1 \; . \qquad (4.4)$$

The parameters are taken to be

$$\left.\begin{aligned} s_0 &= s(0) = 0.25 \\ k_L &= 2.0 \\ k_R &= 1.0 \\ L &= 4(k_L - k_R) = 4.0 \\ t_1 &= -2\ln(s_0) \approx 2.77 \end{aligned}\right\} \qquad (4.5)$$

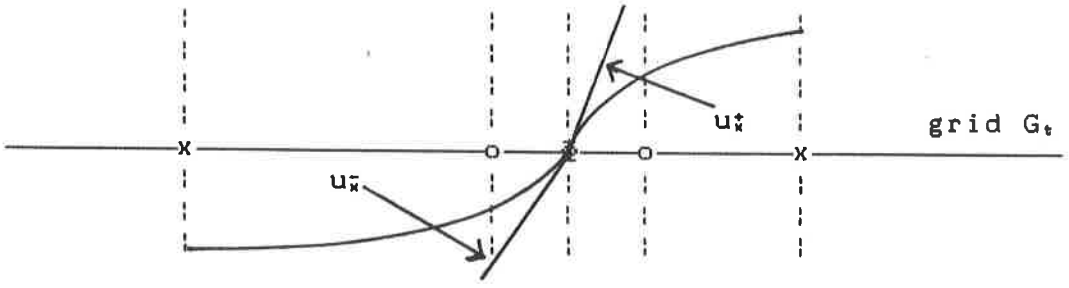
The above problem has the simple analytic solution

$$\left.\begin{aligned} u(x,t) &= x^2 - s_0^2 e^t \\ s(t) &= s_0 e^{t/2} \end{aligned}\right\} \qquad (4.6)$$

4.1.2 Estimation of the Boundary Velocity

In order to be able to track the moving boundary on the grid M_t it is important to be able to estimate its speed $\dot{s}(t)$ from equation (4.3), where u_x^- and u_x^+ are the slopes either side of the boundary. Initially these quantities where taken as the slopes of the piecewise linear approximation in the elements either side of the boundary, but after several experimental runs it was found better to introduce local quadratics in the two elements either side of the boundary and estimate the slope u_x^- and u_x^+ from these (see Fig.4.2).

Fig.4.2. Diagram showing quadratics fitted either side of boundary position, in order to estimate u_x^- and u_x^+



4.1.3 Initial Nodal Distributions on F_o and M_o

Initially their are N_M (N_M even) elements on grid M_t which are placed equally either side of the boundary node with nodal spacing $\mathcal{E}$. We define Ω as the distance initially spanned by the elements on M_o, ($\Omega = \mathcal{E} \times N_M$) and confine the number of

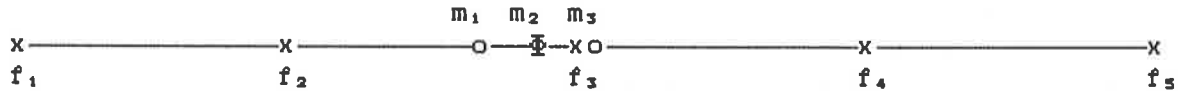
elements N_F on grid F_0 such that

$$N_F < (b-a)/2\Omega \ , \qquad (4.7)$$

where (b-a) is the length of the solution domain, (i.e. the magnitudes of the elements on grid F_0 are at least 2Ω).

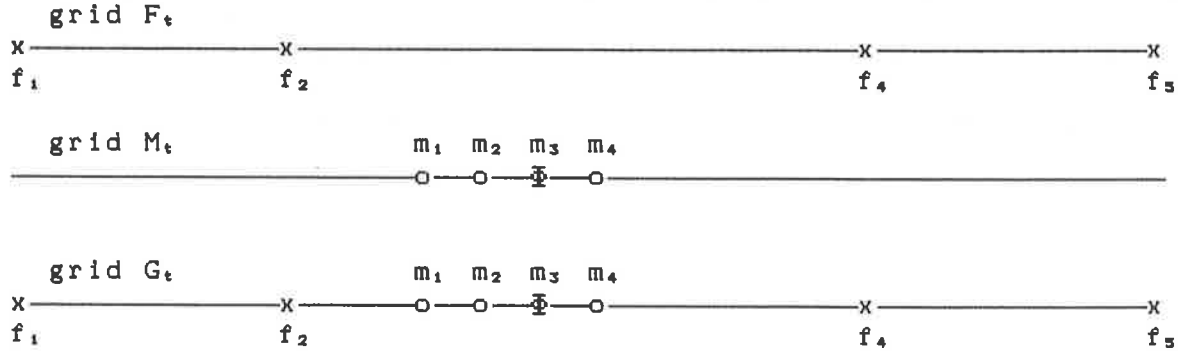
If the minimum distance between the nodes on F_0 & M_0 > $\mathfrak{E}$, then $F_t \equiv F_0$ & $M_t \equiv M_0$, and we are in Mode 0. However if this not the case (see Fig.4.3) we must initially regrid F_t and M_t.

Fig.4.3 Diagram showing that the distance between f_3 & m_3 is less than $\mathfrak{E}$



Constraint (4.7) ensures that at most one node from grid F_0 is within $\mathfrak{E}$ of a node on grid M_0. This being the case, we remove the node from grid F_t and place it in grid M_t at a distance $\mathfrak{E}$ behind node m_1, then renumber the nodes on grid M_t (see Fig.4.4): now we are in Mode -1.

Fig.4.4 Diagram showing initial grids F_t, M_t & G_t if starting mode is Mode -1



4.1.4 Implementation of the NR and ND Algorithms

With the initial grids F_t and M_t defined as above, we can fit the initial data at the grid points by use of equation (4.4), and with the boundary velocity at each time step being calculated by the method outlined in section 4.1.2 we can implement the two algorithms. However we must note that, in step 5 of the NR algorithm, the moving boundary condition (u(s)=0) applies to the node which will represent the boundary at the end of the time step. If initially we are in Mode -1 then the NR algorithm is started at step 7, and the ND algorithm is started at step 5, otherwise both are started at step 1.

Both algorithms were run to a time of t=2.5, with the following data set;

$N_M = 2$, $N_F = 22$, $\varepsilon = 0.01$, $dt = 10^{-5}$,

the results for the NR algorithm being shown in Fig.4.5 and those for the ND algorithm in Fig.4.6. As can be seen, both methods give very good results, with the % error in the location of the boundary at the final time t being approximately 0.001% for both methods. However, as we were using an explicit time-stepping scheme, $dt < \varepsilon^2$ for stability, and as can be seen from the results both runs took a great deal of cpu time (≈1625 secs.). Since it is hoped to try these methods on more advanced problems and also perhaps extend the ideas to 2-d, we decided to construct an implicit version of the algorithm in the hope of decreasing the cpu time.

4.1.5 An Implicit Version of the NR and ND Algorithms.

The main objective in constructing an implicit algorithm is to inhibit node overtaking and allow larger time steps to be taken. In this problem the boundary velocity can easily be expressed explicitly in terms of the nodal amplitudes (v_i) and positions (s_i), which makes it possible to solve for the boundary position s at the new time step. However doing this can lead to node overtaking and hence disrupt the solution (see Fig.3.6). Miller & Miller (see[5]) overcome this problem by use of penalty functions, but this makes the method quite complicated and also involves problem dependent tuning parameters. We decided to use a Predictor-Corrector iterative method for time-stepping the nodes, with an implicit Euler time-stepping method for the nodal amplitudes.

The NR & ND algorithms are as before except for the steps which involve time-stepping, which are now carried out as follows.

4.1.6 Time-Stepping Using the Implicit Algorithm

1) Calculate the explicit velocity of the boundary at time level n, $\dot{s}^n$ and use this as a first estimate to the implicit velocity $\dot{s}^{n+1}$. (i.e. set $\dot{s}^{n+1} = \dot{s}^n$)

2) If necessary adjust the time step dt, so as to avoid nodal spacings becoming < δ or if a node from grid M_t has to be dropped off onto grid F_t.

3) Calculate the nodal positions at time level n+1 (note that step 2 avoids node overtaking), i.e. $s^{n+1} = s^n + dt.\dot{s}^{n+1}$

4) Using the above speed, nodal positions and dt, calculate the solution of the Galerkin equations (2.12) at time level n+1 using implicit Euler time-stepping. (i.e. still solve $A\underline{v} = \underline{b}$ as equations are linear)

5) Using the solution at time level n+1 calculate the implicit velocity $\dot{s}^{n+1}$.

6) Repeat from step 2 until the implicit velocity $\dot{s}^{n+1}$ converges to a certain tolerance.

4.1.7 Implementation of the Implicit Algorithms. with Error Measurements.

The grids are initialised as in the explicit case (see section 4.1.3). The boundary velocity is calculated by the method outlined in section 4.1.2. The first runs using the implicit algorithms were done using a similar data set to that used in the explicit method;

$N_M = 2$, $N_F = 22$, $\varepsilon = 0.01$, $dt = 10^{-2}$,

the only difference being that dt is increased by a factor of 10^3. The results for the two algorithms can be seen in Figs.4.7 & 4.8. In both cases the error in boundary position at the final time is less than 0.3%. Although this is worse than the explicit error, the cpu time used has been reduced to an average of 10.5 secs, this being a decrease of $\approx$ 99%.

Next it was decided to decrease the number of nodes on the fixed grid F_o keeping the other parameters the same. First N_F was decreased to 10 (see Figs.4.9 & 4.10) and then to 5 (see Figs. 4.11 & 4.12). As can be seen from the results, the moving boundary was accurately tracked (average error of 0.29%) with cpu times going as low as 7.5 secs.

For an accuracy test it was decided to include some error measurements for both methods. The data chosen was as above with N_F = 10 (Figs. 4.11 & 4.12) and it was decided to first measure the absolute error between the numerical and exact boundary velocity, for ten iterations before and after the first node reversal/deletion (n-10,...,n,...,n+10). At time step n the velocity calculated is the first velocity after node reversal/deletion has taken place. The error for the NR algorithm can be seen in Fig.4.13 with the ND error in Fig.4.14. It is interesting to note that after the node reversal/deletion iteration, the error actual decreases (quite considerably for the NR algorithm), although the overall pattern is that the error gradually increases, as might be expected since the implicit method tends to overestimate the position of the boundary (see appendix).

Finally it was decided to measure the relative L_2 error (see Heath [10]) between the exact and numerical solution for ten iterations before and after the first node reversal/deletion (n-10,...,n,...,n+10). The results for the NR and ND algorithm are shown in Figs.4.15 & 4.16 respectively. Here it is worth noting that although the error slightly increases during node reversal/deletion, the general trend is of a decreasing L_2 norm.

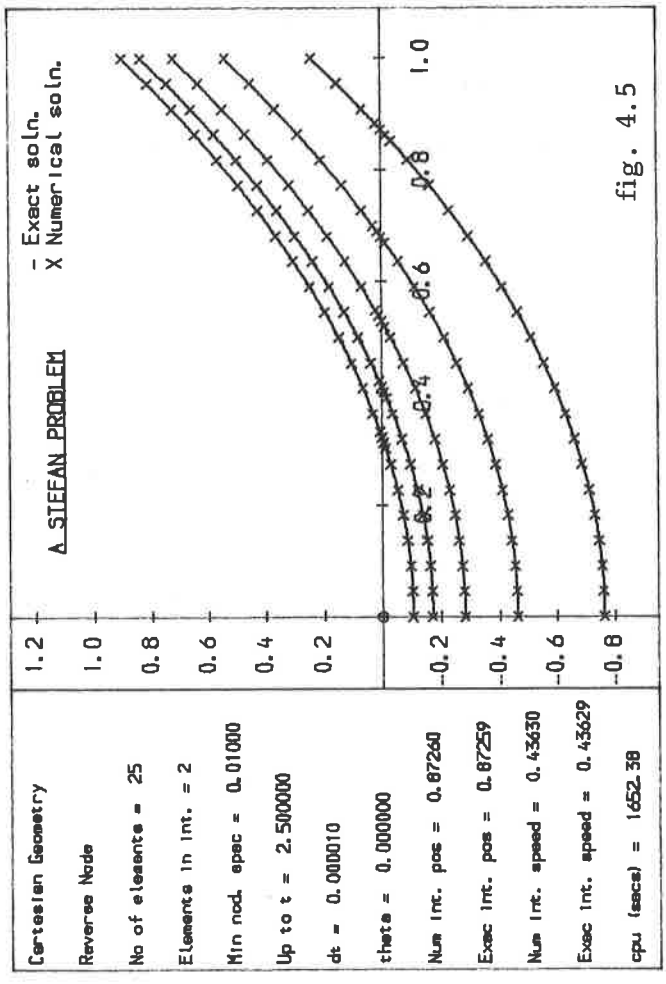


fig. 4.5

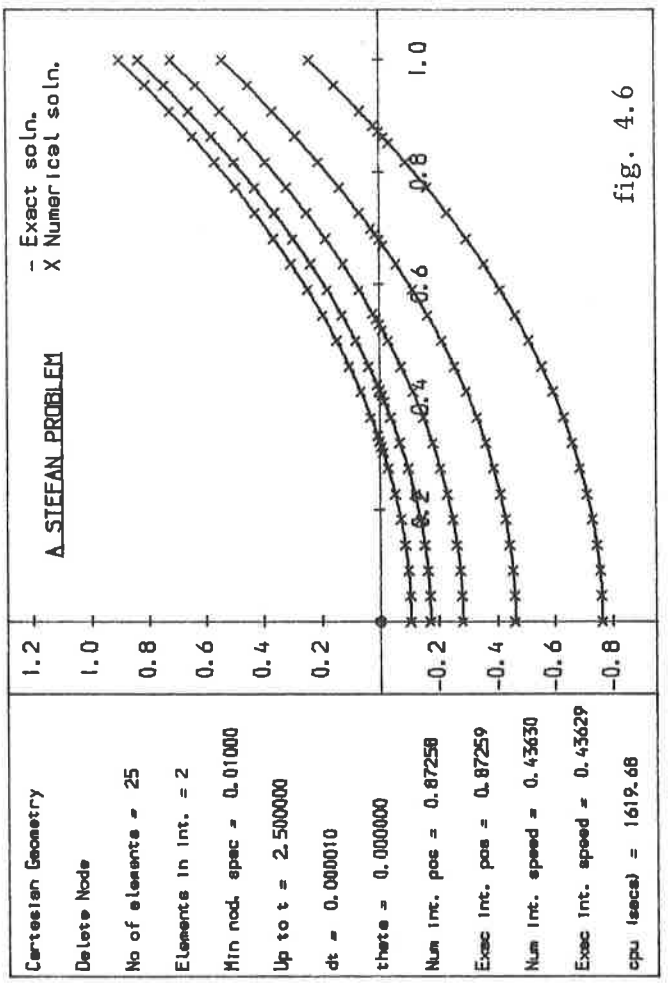


fig. 4.6

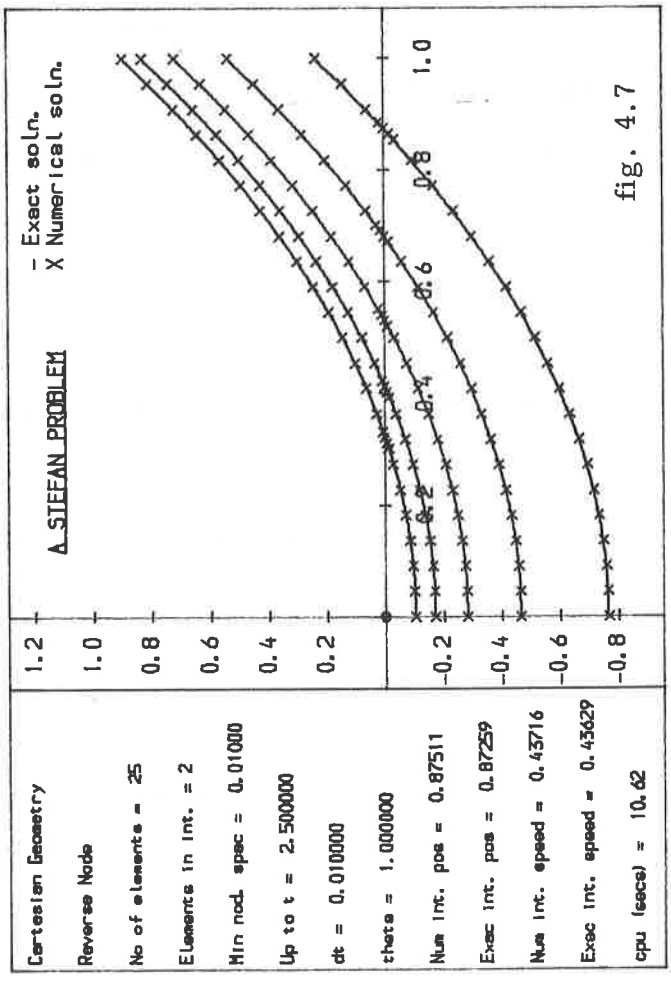


fig. 4.7

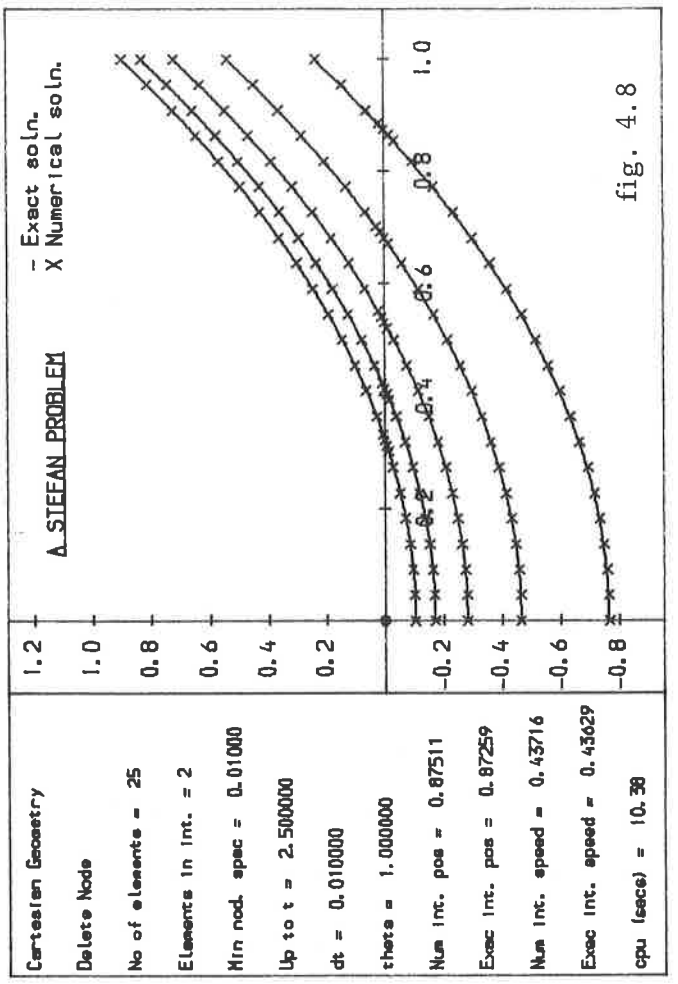


fig. 4.8

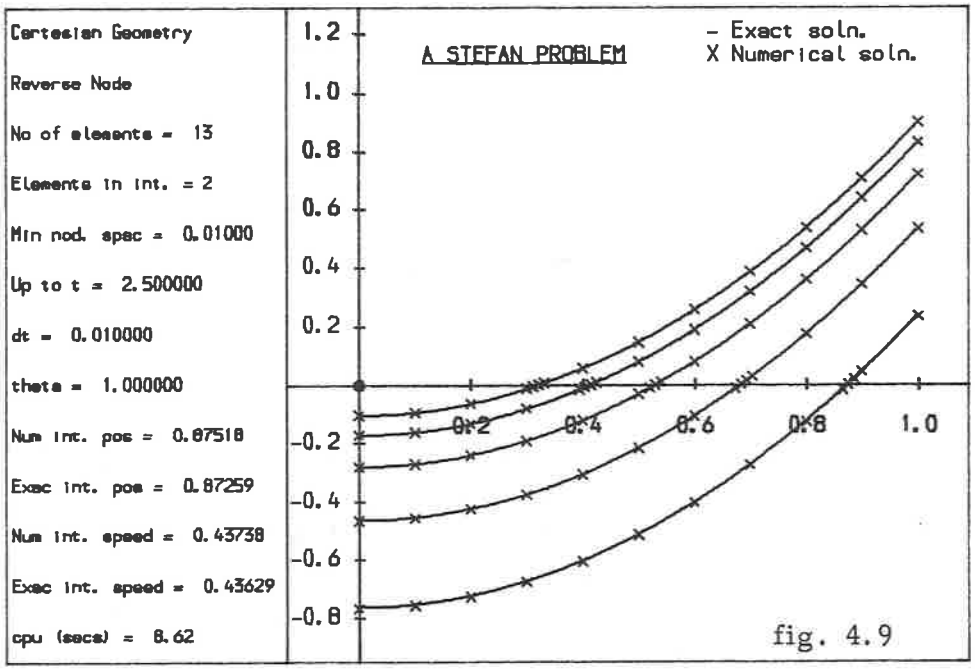


fig. 4.9

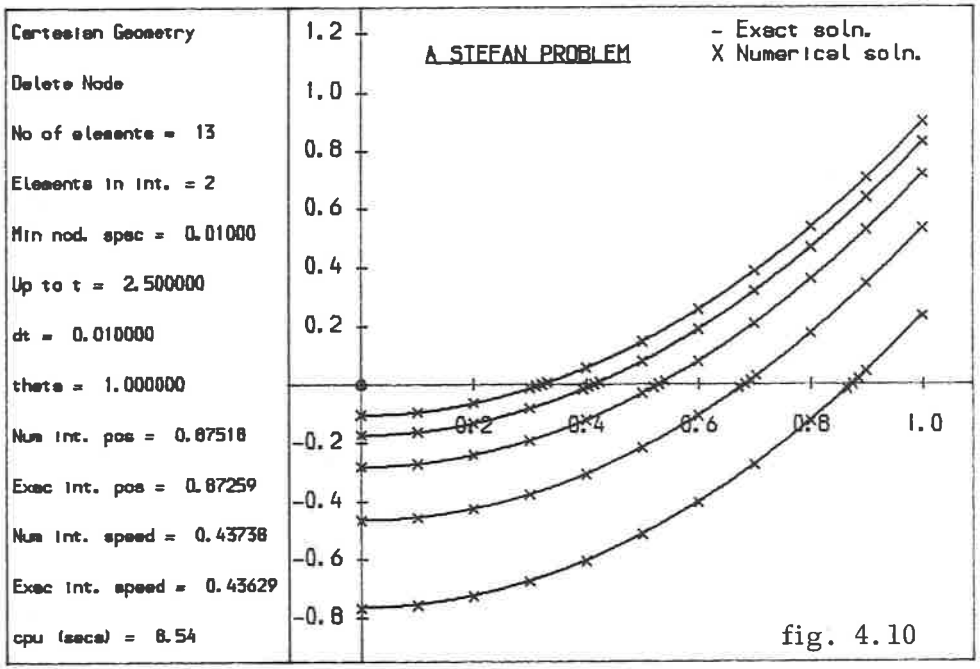


fig. 4.10

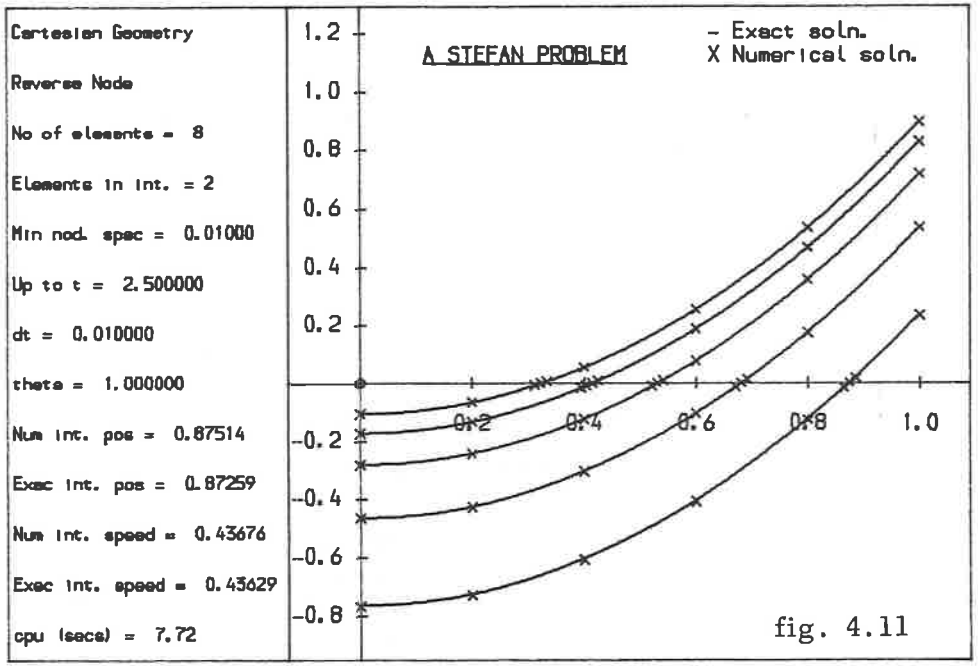


fig. 4.11

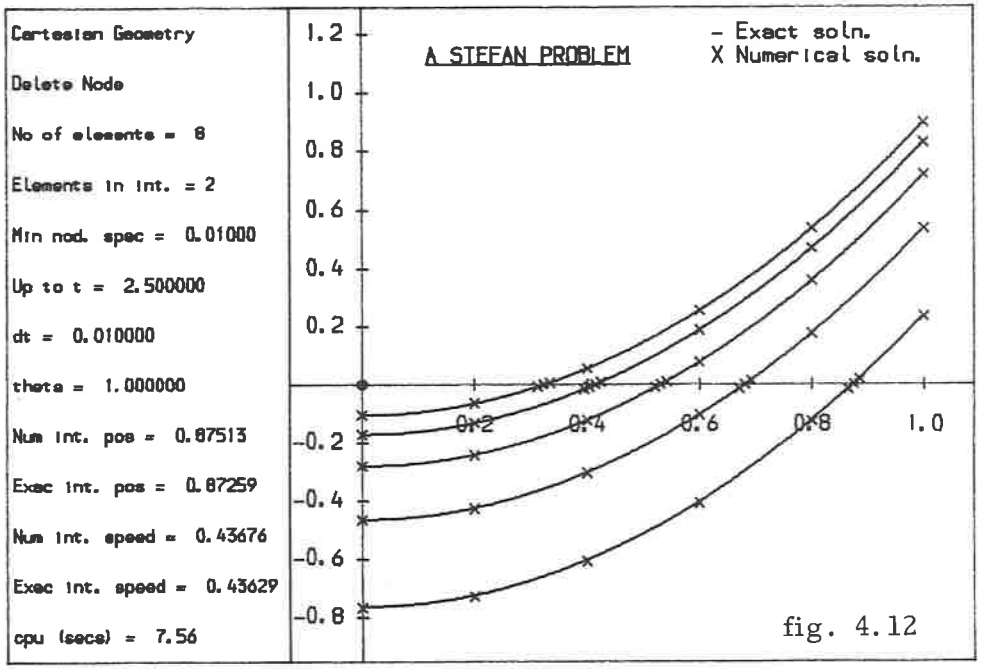


fig. 4.12

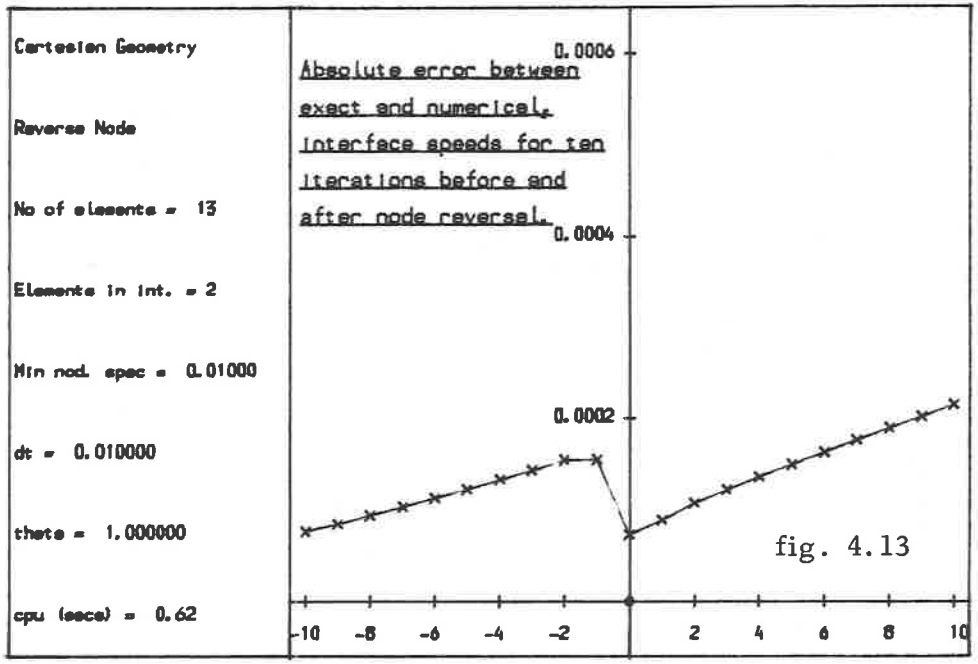


fig. 4.13

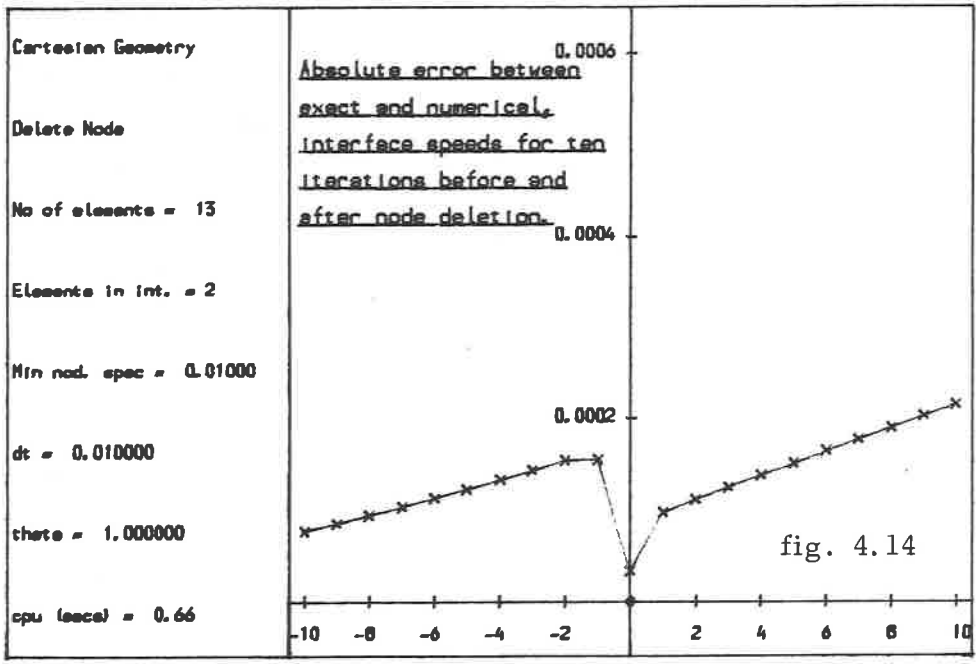


fig. 4.14

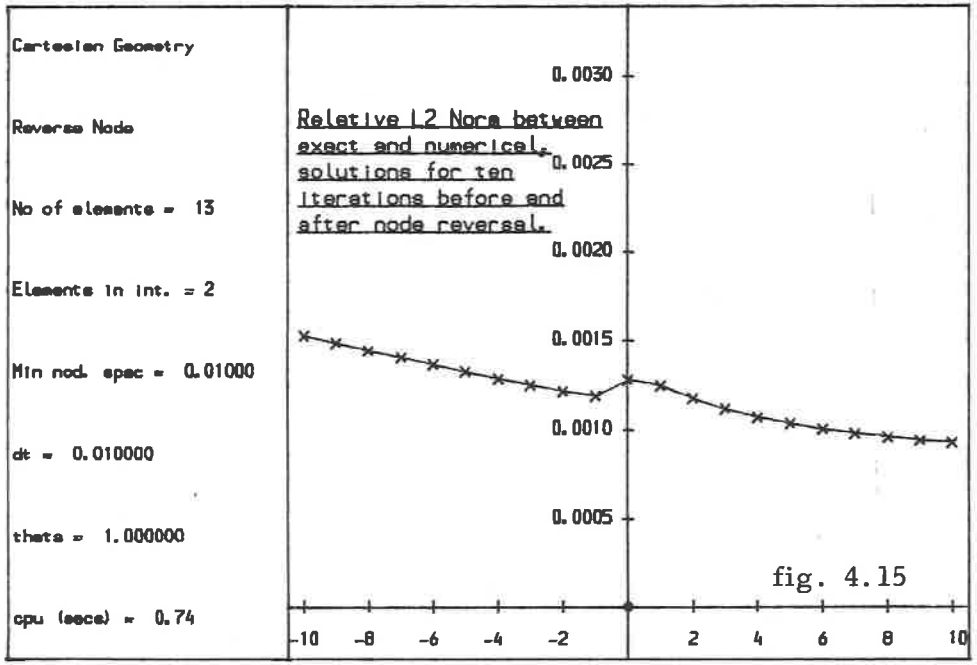


fig. 4.15

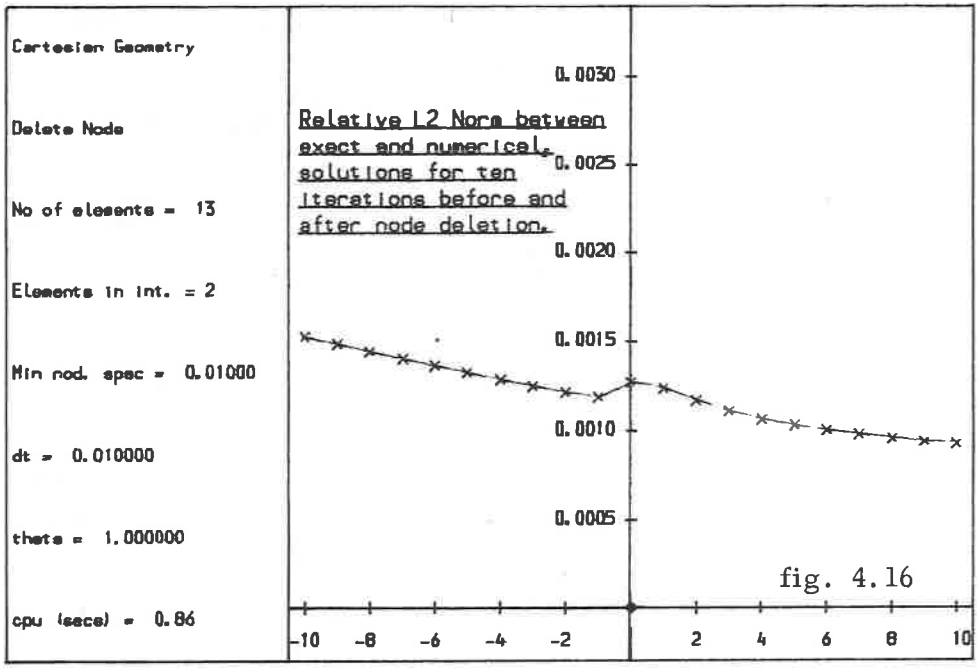


fig. 4.16

4.1.8 The Radial Moving Boundary Problem.

Since the eventual aim of our work is application to oil reservoir modelling, it is appropriate to consider a radial symmetrical version of the previous moving boundary problem, as reservoirs are often taken to be radially symmetric. The problem is similar to the cartesian one and can be stated in the form

$$\left.\begin{aligned} u_t &= k_L\,\mathrm{del}^2 u - s_0^2 e^t - 4k_L \qquad a < r < s(t) \\ u_t &= k_R\,\mathrm{del}^2 u - s_0^2 e^t - 4k_R \qquad s(t) < r < 1 \end{aligned}\right\} \; 0 < t < t_1 \qquad (4.8)$$

where $\mathrm{del}^2 = \frac{1}{r}\frac{d}{dr}\left(r\frac{d}{dr}\right)$

where s(t) is the position of the moving boundary.

The fixed boundary conditions are

$$\left.\begin{aligned} u_r &= 2.a \qquad r = a \\ u_r &= 2 \qquad r = 1 \end{aligned}\right\} \; 0 < t < t_1 \qquad (4.9)$$

and the boundary conditions at the moving boundary are

$$\left.\begin{aligned} u &= 0 \\ k_L u_r^- - k_R u_r^+ &= L\dot{s}(t) \end{aligned}\right\} \; 0 < t < t_1 \qquad (4.10)$$

with initial conditions

$$u(r,0) = r^2 - s_0^2 \qquad 0 < x < 1 \qquad (4.11)$$

and parameters

$$\left.\begin{aligned} s_0 &= s(0) = 0.25 \\ k_L &= 2.0 \\ k_R &= 1.0 \\ L &= 4(k_L - k_R) = 4.0 \\ t_1 &= -2\ln(s_0) \approx 2.77 \end{aligned}\right\} \qquad (4.12)$$

Implicit solutions to the above problem using both the NR and ND algorithms were constructed.

In this case we replace the inner product $\langle \cdot , \cdot \rangle$ by

$$\langle \cdot , \cdot \rangle = \int_a^1 r \, . \, \cdot \, . \, \cdot \, dr \qquad (4.13)$$

Which ensures conservation, also the equation exhibits a singularity at the origin and therefore the solution domain is taken to be [a,1], where a is chosen as a = 0.1.

The numerical parameters initially used were the same as for the first implicit case on the cartesian geometry, namely,

$N_M = 2$, $N_F = 22$, $\varepsilon = 0.01$, $dt = 10^{-2}$.

The results for the NR algorithm can be seen in Figs.4.17, with those for the ND in Fig.4.18. It may be noted that the error between the exact and numerical boundary position at time t=2.5 is slightly less for this case (0.08%) than the cartesian one, although as can be seen in Figs.4.19 & 4.20, when N_F is reduced to 10, the numerical boundary location

begins to trail behind the exact one, with an error of up to -3.8% when N_f = 5 (see Figs.4.21 & 4.22)

A reason why the accuracy of the numerical boundary decreasse with N_F for the radial problem and not the cartesian, may be attributed to the "area" represented by each element. For equally spaced elements on the cartesian grid the areas represented by each elements are equal, but for the radial case the area increases with r. It was therefore decided to insert an extra element near the outer boundary (r=1) on the fixed grid F_0. Doing so increased N_F to 6, but the resulting error in numerical boundary location was reduced to -0.71% (see Figs.4.23 & 4.24).

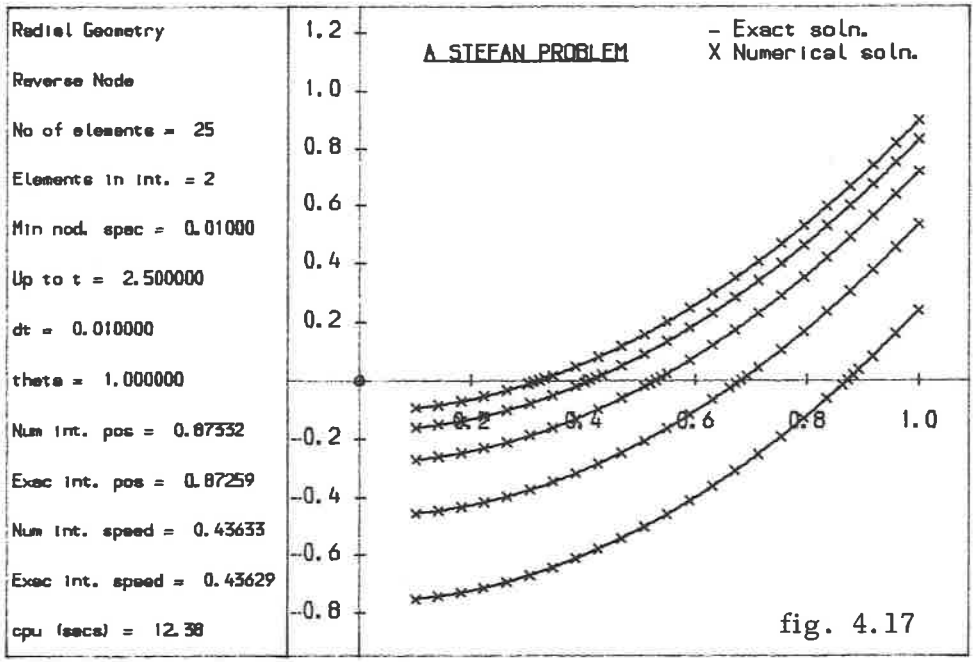


fig. 4.17

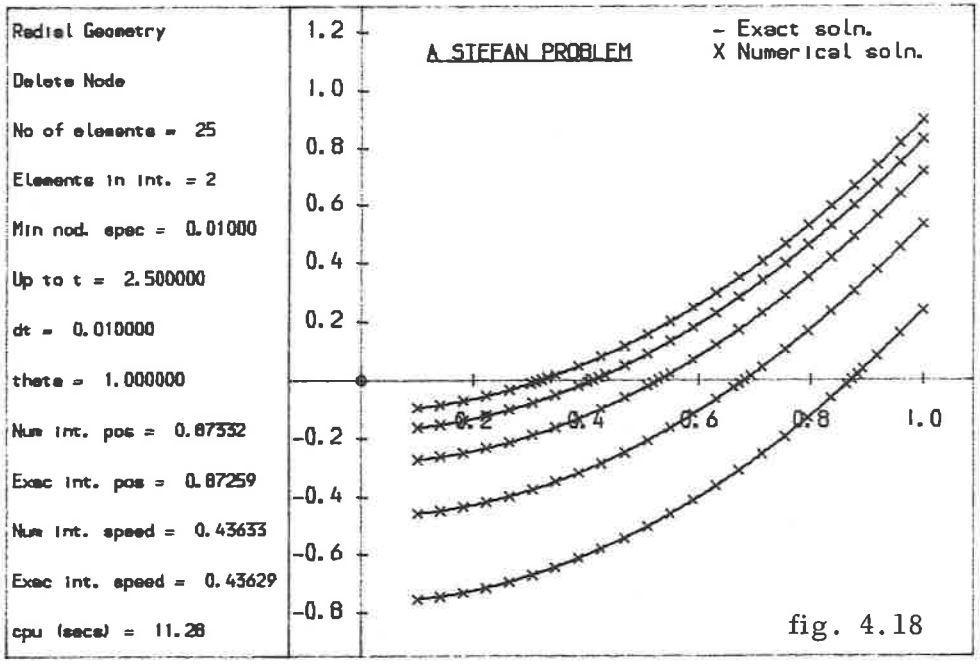


fig. 4.18

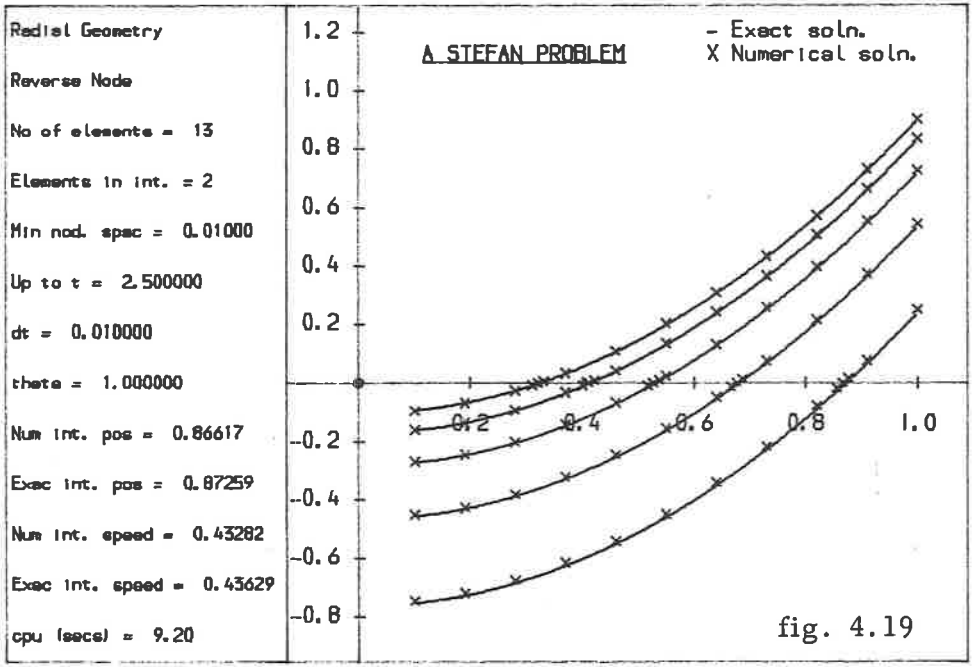


fig. 4.19

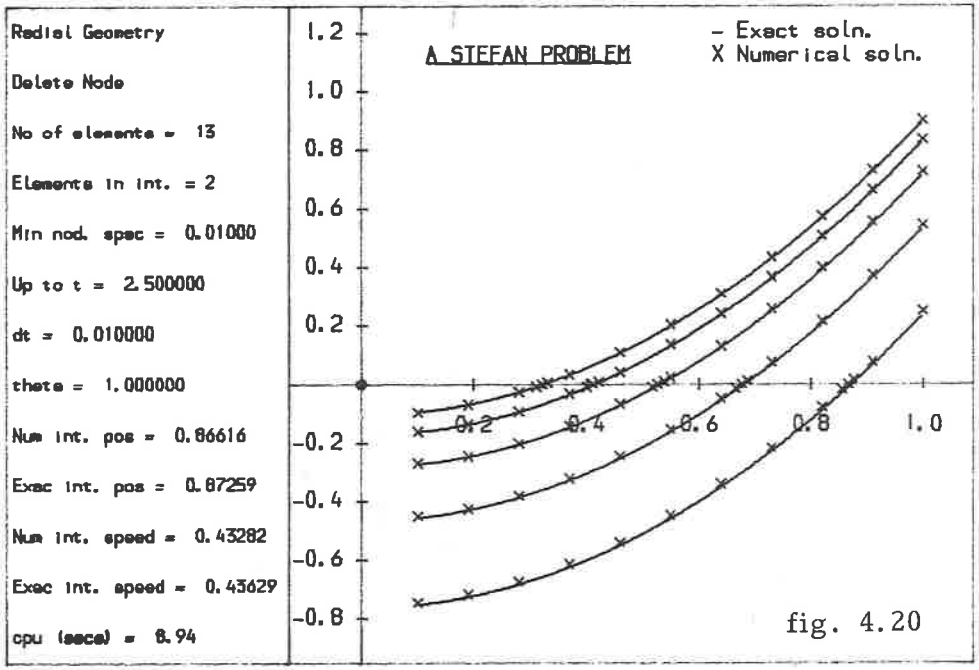


fig. 4.20

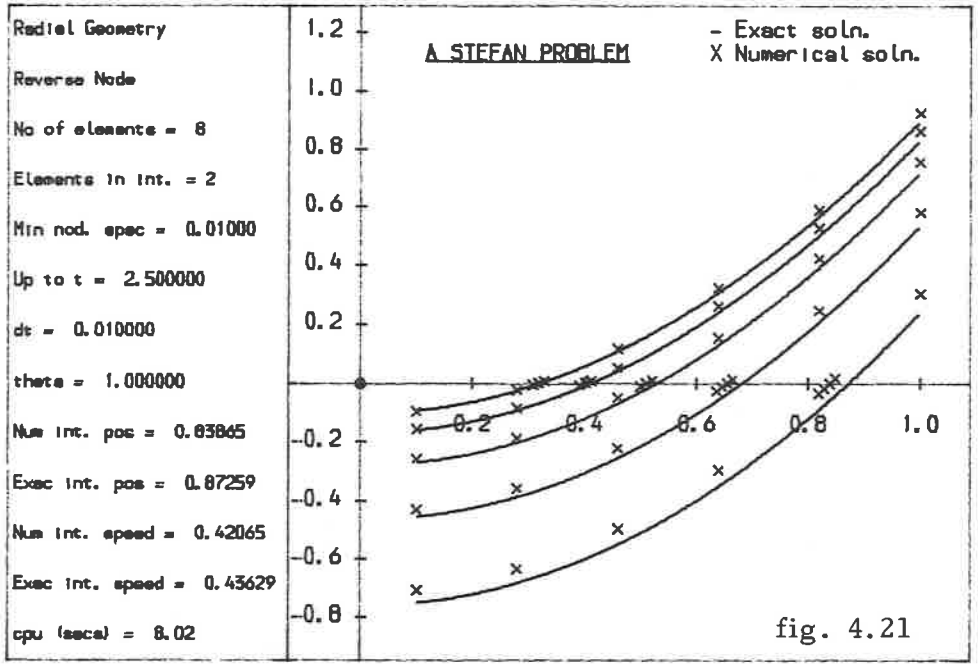


fig. 4.21

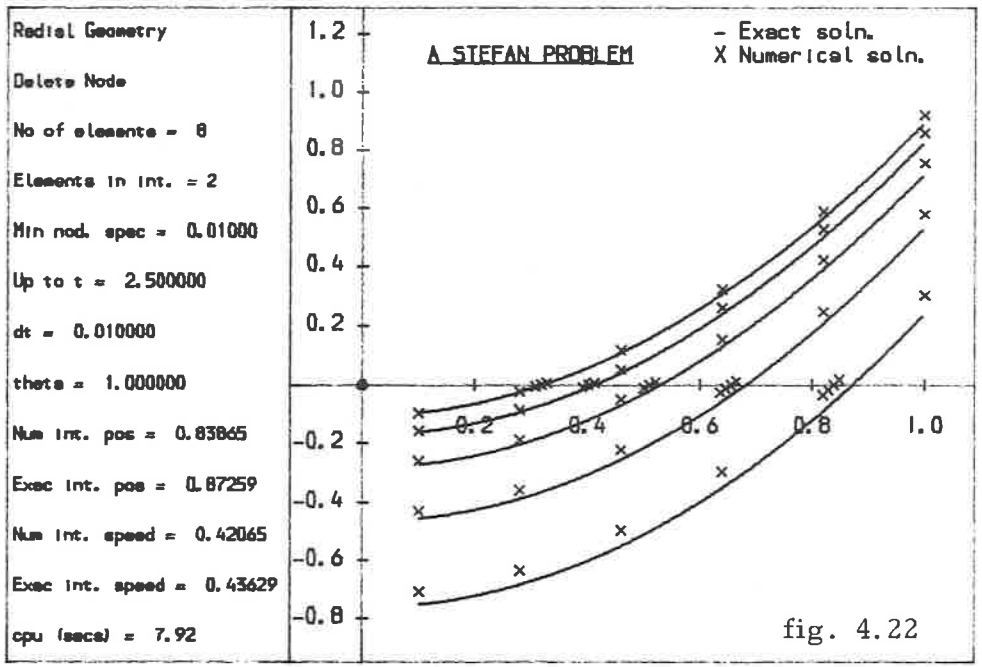


fig. 4.22

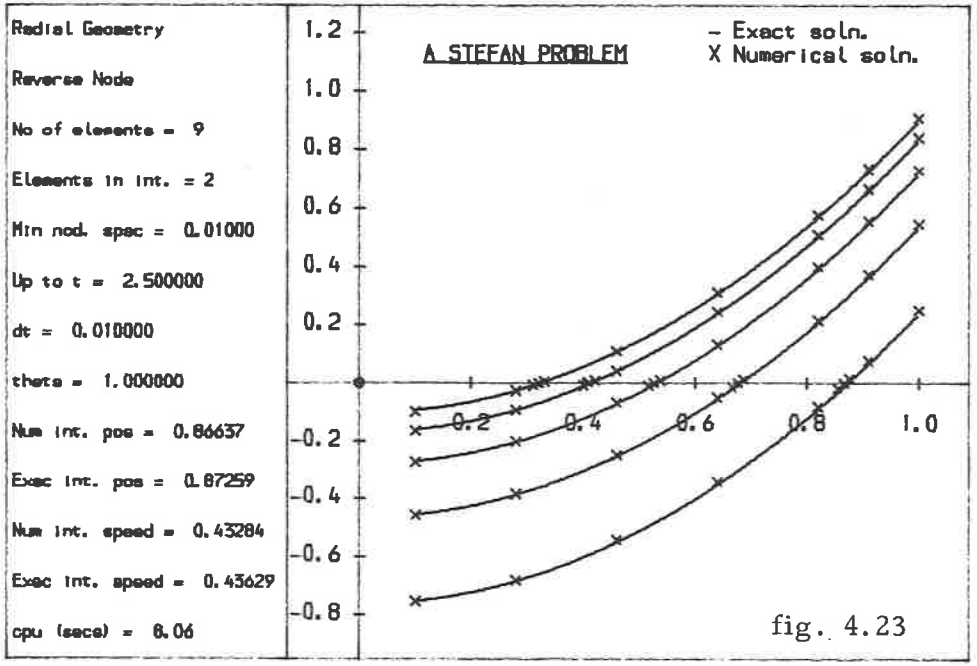


fig. 4.23

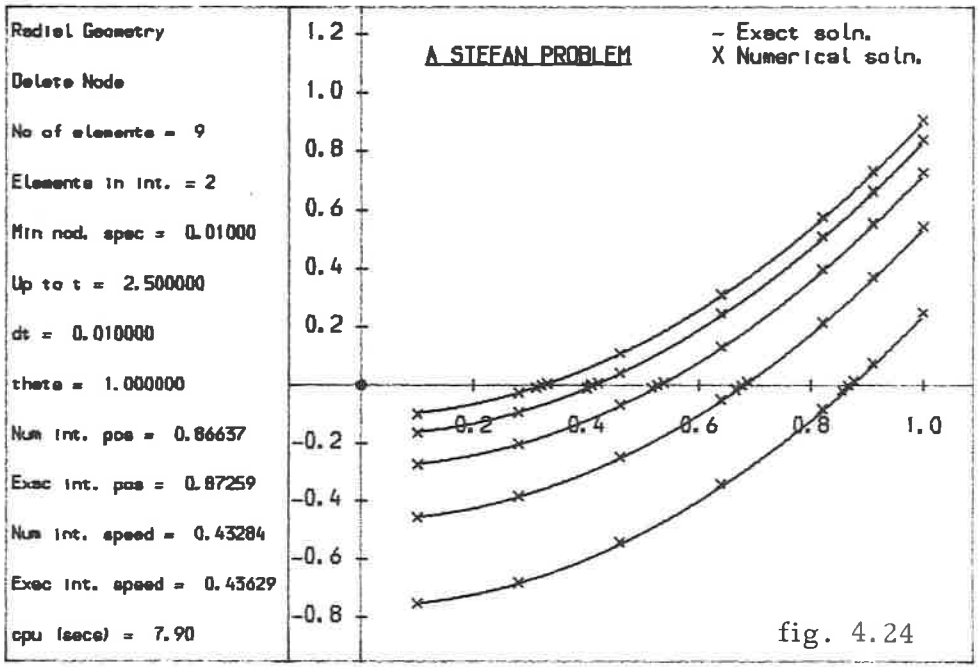


fig. 4.24

4.2 Propagation of Steep Fronts.

It was decided to try the NR & ND algorithms on a more demanding problem than those described in the last section, as those problem had the advantage of having the value at the moving boundary pre-determined (u(s)=0). The propagation of a steep front in the problem, with the requirement that the speed of the front can still be estimated in some form, suggested Burgers' equation which is considered below.

4.2.1 Burgers' Equation

Burgers' equation is an advection diffusion equation of the form

$$u_t + uu_x = \varepsilon u_{xx} \qquad x \in [0,b] \tag{4.14}$$

This equation has a suitable exact solution (see Johnson [11]) of the form

$$u(x,t) = f(\sigma) \tag{4.15}$$

$$\sigma = x - \mu t - \beta \tag{4.16}$$

$$f(\sigma) = \frac{\{ (\mu+\alpha) + (\mu-\alpha)\exp(\frac{\alpha\sigma}{\varepsilon}) \}}{\{ 1 + \exp(\frac{\alpha\sigma}{\varepsilon}) \}} \tag{4.17}$$

Boundary and initial conditions are given by the analytic solution (4.17) as

$$u(x,0) = f(x - \beta) \tag{4.18}$$

$$u(0,t) = f(-\mu t - \beta) \tag{4.19}$$

$$u(b,t) = f(b - \mu t - \beta) \tag{4.20}$$

The arbitrary constants in (4.17) are chosen as $\alpha = 0.4$, $\beta = 0.125$, $\mu = 0.6$. For these parameters the initial data represents a steep front centred at β between the states 1.0 and 0.2 with the steepness of the front depicted by ε. The exact solution represents a travelling wave solution, the speed of propagation being $\mu = 0.6$.

4.2.2 Numerical Representation of Front & Front Speed

10 elements were used to represent the front on grid M_t and, since the initial position of the front was known, elements were placed at a spacing δ either side of it. The spacing δ chosen was dependent on the steepness of the front. (i.e. ε) After several experiments it was found to be best to use a value of 3ε. In order to estimate the speed of the front it is important to know the value of u on both sides of the front and therefore, as the front is being represented by the nodes on grid M_t, it is important to know the number of the first and last node, i.e. in the present case 1 and 10 or 11 (depending on the Mode). For convenience we will call these numbers iff (i first front) and ilf (i last front) respectively. The initial distribution of nodes on grid F_0 is the same as described in section 4.1.3. Finally before any

runs can take place the actual speed of the front has to be estimated and since we are dealing with a very steep front (width $\approx 30\varepsilon$ where $\varepsilon \approx 10^{-4}$), we assume that it has a shock structure with the values u(iff) an u(ilf) representing the states on either side. We then apply the Rankine-Hugoniot shock condition to the advective part of equation (4.14) in order to estimate the speed of the front, i.e.,

$$s = \frac{\frac{1}{2}(u(iff)^2 - u(ilf)^2)}{u(iff) - u(ilf)} = \frac{1}{2}(u(iff) - u(ilf)) \qquad (4.21)$$

4.2.3 Numerical Results for Burgers' Equation Using Numerical Newton-Raphson

Using the methods outlined in the previous section to initialise the grids (F_o & M_t) and calculate the speed of propagation of the front, several runs were carried out using the implicit version of the algorithm, as described in sections 4.1.5 & 4.1.6. However, as the differential equation is now non-linear, the Galerkin equations produce a set of non-linear equations which are solved using a general Newton-Raphson technique, where the Jacobian is calculated numerically.

The initial numerical data set was

$N_M = 10$, $N_F = 5$, $dt = 10^{-2}$, $\varepsilon = 10^{-2}$, $\varepsilon = 3\varepsilon$,

The results for the NR algorithm are shown in Fig.4.25 and those for the ND algorithm in Fig.4.26. Very slight oscillations occur in the solution, of the order of 0.03% (u(ilf) = 1.00259 instead of 1.0 in Fig.4.26), but the overall measure of error (L_2 norm) at time t = 5.0 is very small being

only 3.1×10^{-3}. It was then decided to use the same, data set, but with dt increased to 0.1, results for which can be seen in Figs.4.27 & 4.28. The point to note here is that for the NR algorithm the L_2 norm increases by just 0.00045, whereas the cpu time decreases from 24 secs. to 9 secs.: for the ND algorithm the L_2 norm actually decreased as the time step increased.

Finally in this sub-section it was decided to decrease ε to 10^{-4} with the data set

$N_H = 10$, $N_F = 5$, $dt = 10^{-2}$, $\varepsilon = 10^{-4}$, $\delta = 3\varepsilon$,

the results of which can be seen in Figs.4.29 & 4.30. Again, very slight oscillations occurred (of the order 0.11%) for both methods, but the overall result was good. This was also the case with $\varepsilon = 10^{-5}$ (see Figs. 4.31 & 4.32) where the data set was

$N_H = 10$, $N_F = 10$, $dt = 10^{-2}$, $\varepsilon = 10^{-5}$, $\delta = 3\varepsilon$,

here the oscillations being of the order 0.13%.

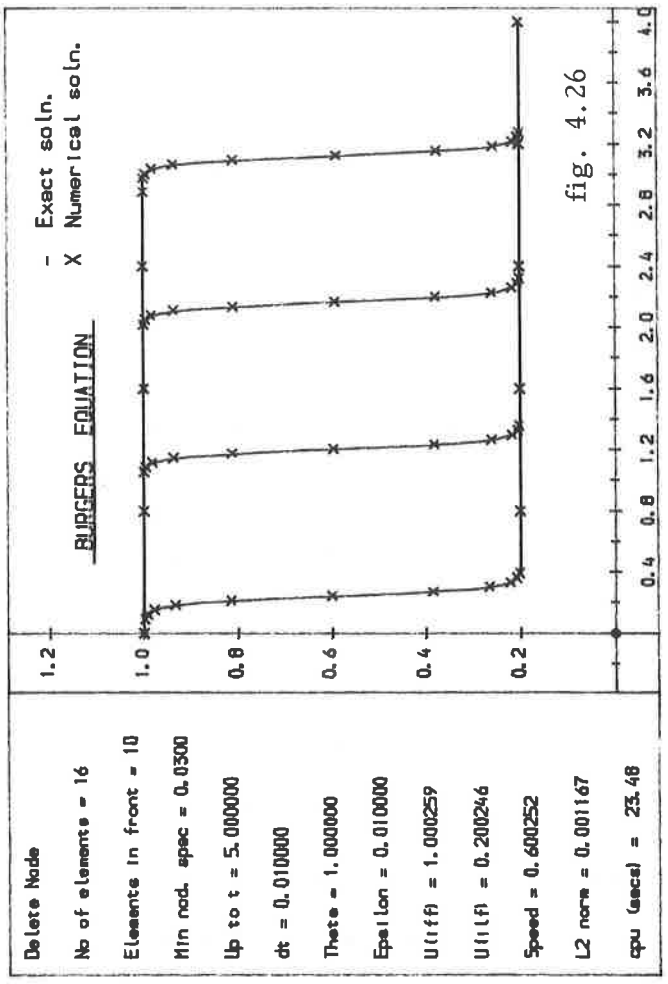


fig. 4.26

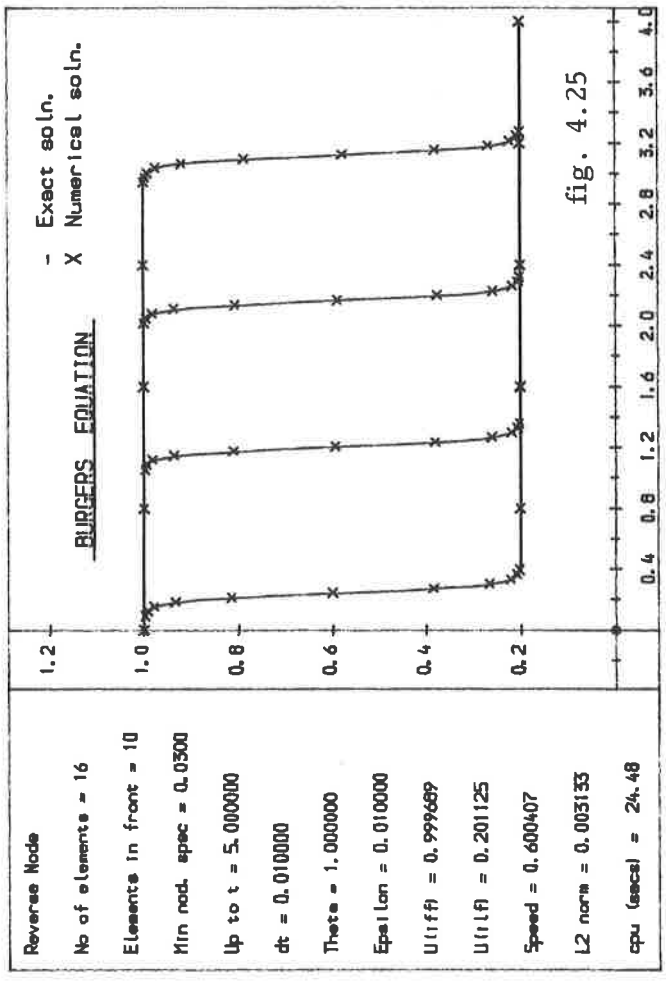


fig. 4.25

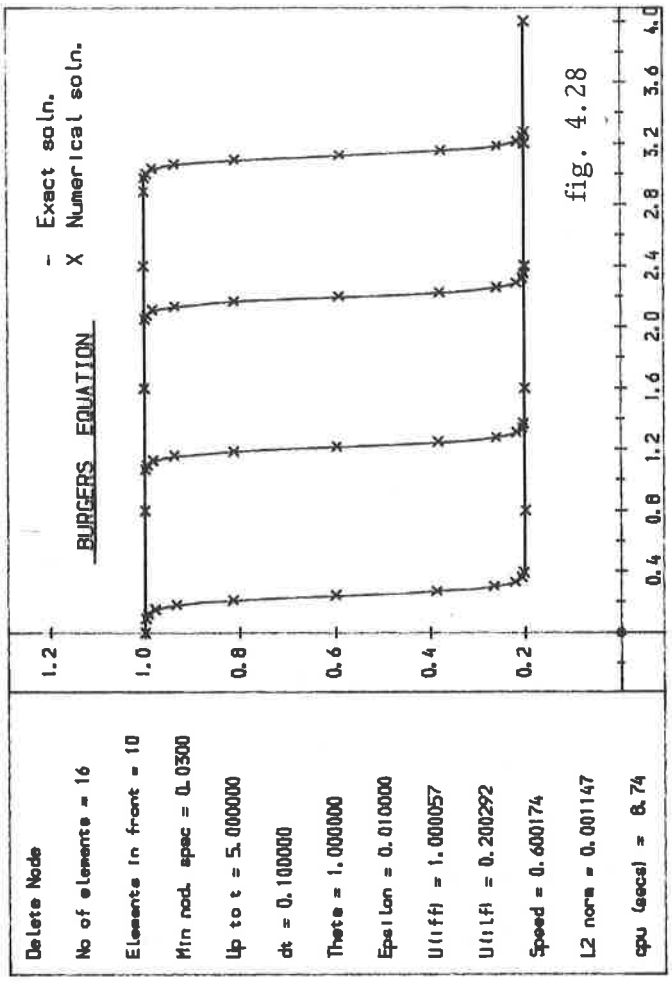


fig. 4.28

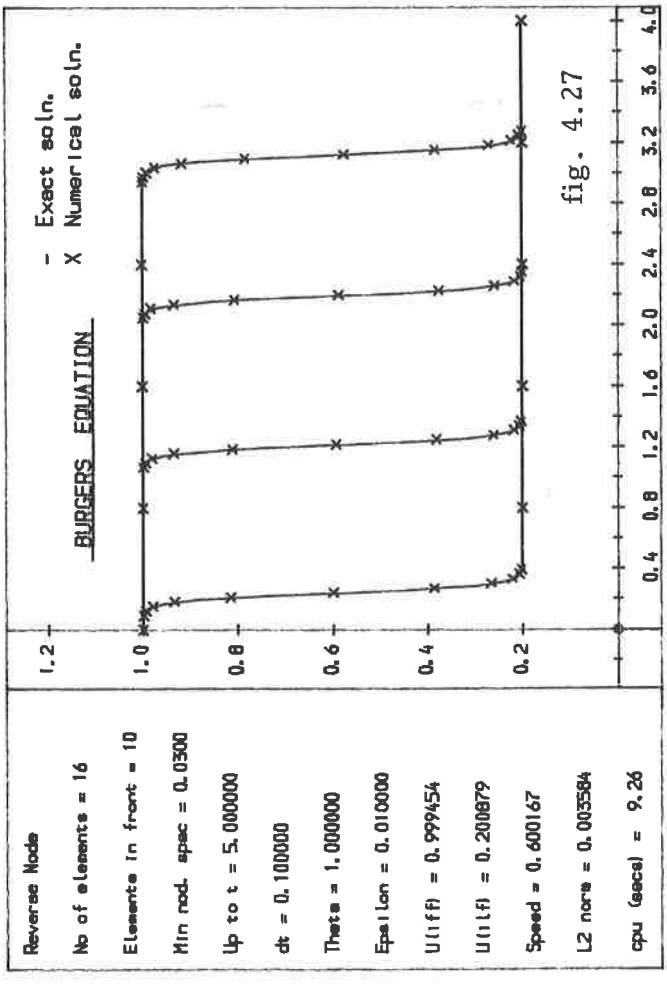


fig. 4.27

Reverse Node

No of elements = 16

Elements in front = 10

Min nod. spec = 0.00030

Up to t = 5.000000

dt = 0.010000

Theta = 1.000000

Epsilon = 0.000100

U(iff) = 0.999949

U(ilf) = 0.202761

Speed = 0.601355

L2 norm = 0.014632

cpu (secs) = 25.40

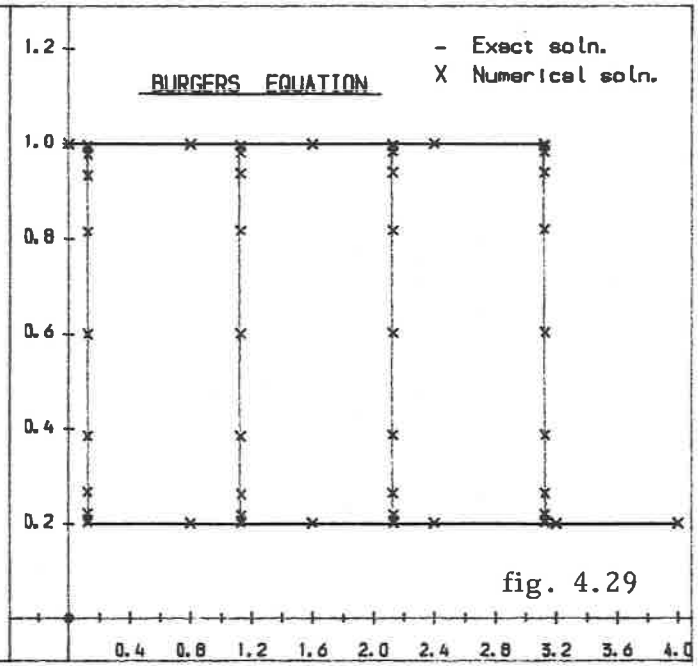


fig. 4.29

Delete Node

No of elements = 16

Elements in front = 10

Min nod. spec = 0.00030

Up to t = 5.000000

dt = 0.010000

Theta = 1.000000

Epsilon = 0.000100

U(iff) = 0.999945

U(ilf) = 0.202762

Speed = 0.601353

L2 norm = 0.014634

cpu (secs) = 24.96

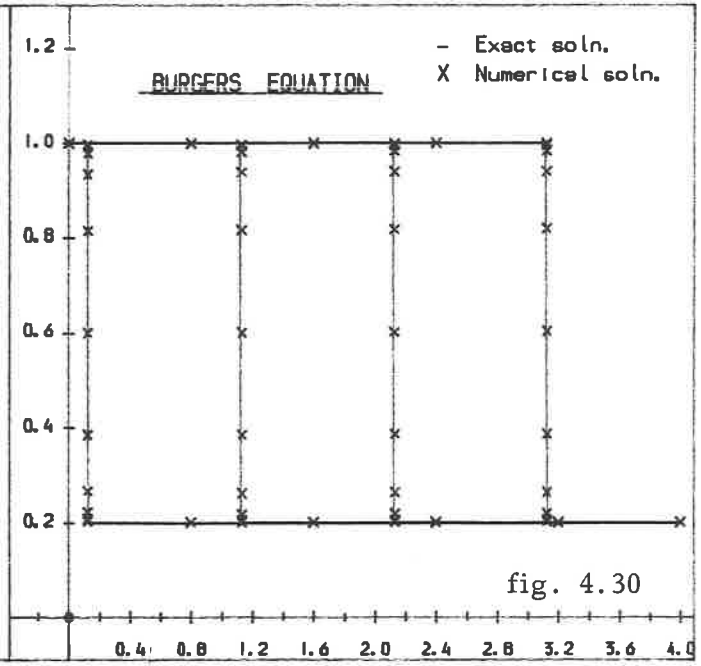


fig. 4.30

Reverse Node

No of elements = 21

Elements in front = 10

Min nod. spec = 0.00003

Up to t = 5.000000

dt = 0.010000

Theta = 1.000000

Epsilon = 0.000010

U(iff) = 0.998900

U(ilf) = 0.202310

Speed = 0.600605

L2 norm = 0.016004

cpu (secs) = 35.18

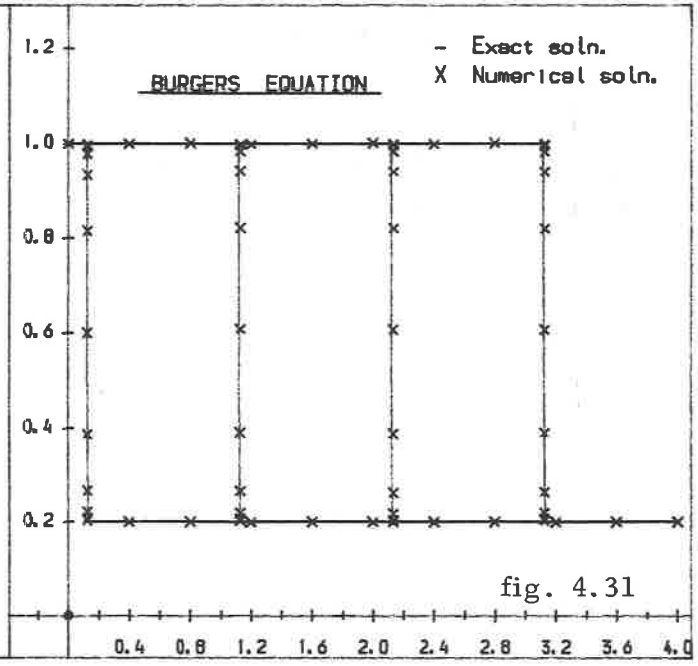


fig. 4.31

Delete Node

No of elements = 21

Elements in front = 10

Min nod. spec = 0.00003

Up to t = 5.000000

dt = 0.010000

Theta = 1.000000

Epsilon = 0.000010

U(iff) = 0.998901

U(ilf) = 0.202310

Speed = 0.600605

L2 norm = 0.016004

cpu (secs) = 33.66

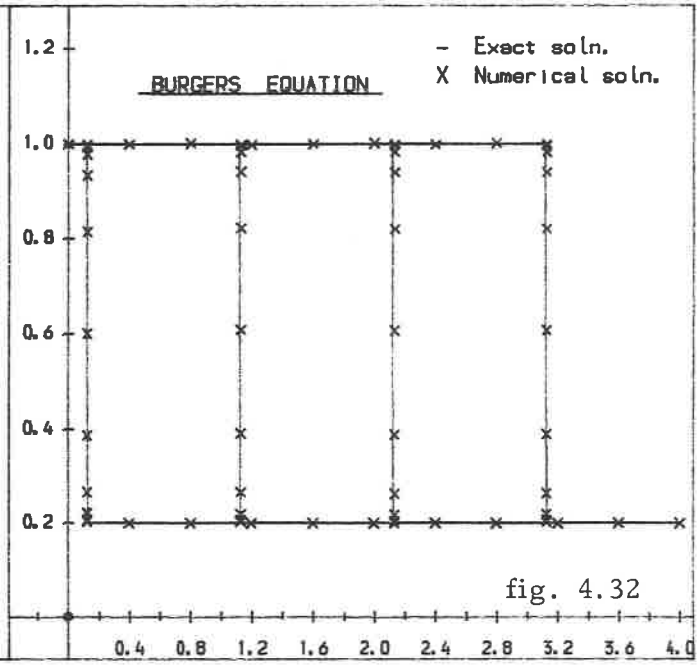
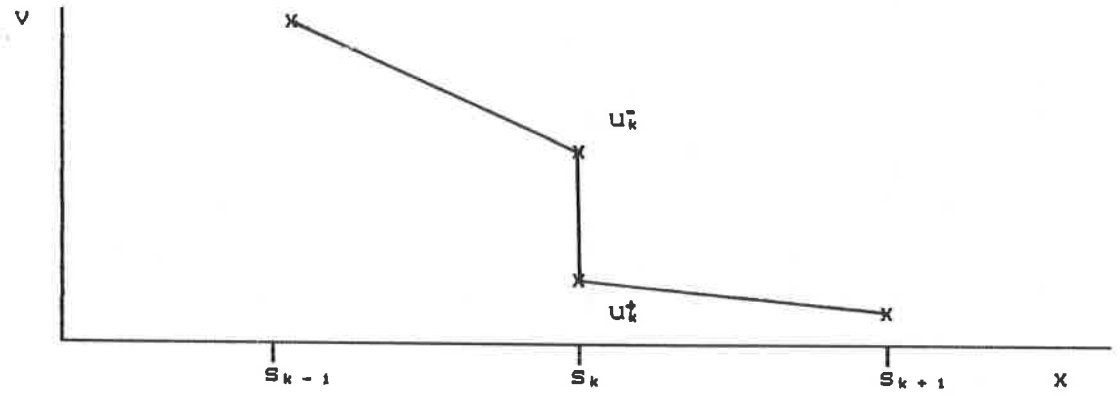


fig. 4.32

4.3 Shock Problems.

In order to represent a shock or contact discontinuity we may use the idea of a point in the solution domain being double valued (see Fig.4.33)

Fig.4.33 Diagram showing double valued point at s_k.



The ND algorithm can be used without amendment on this type of problem, but for the NR algorithm trouble occurs during step 5 (node reversal) when, as in the above case, node s_{k+1} must represent the shock at the end of the time step. This is because at time-level n it is only single valued and it is then required to be double valued at time level n+1.

4.3.1 NR Shock algorithm.

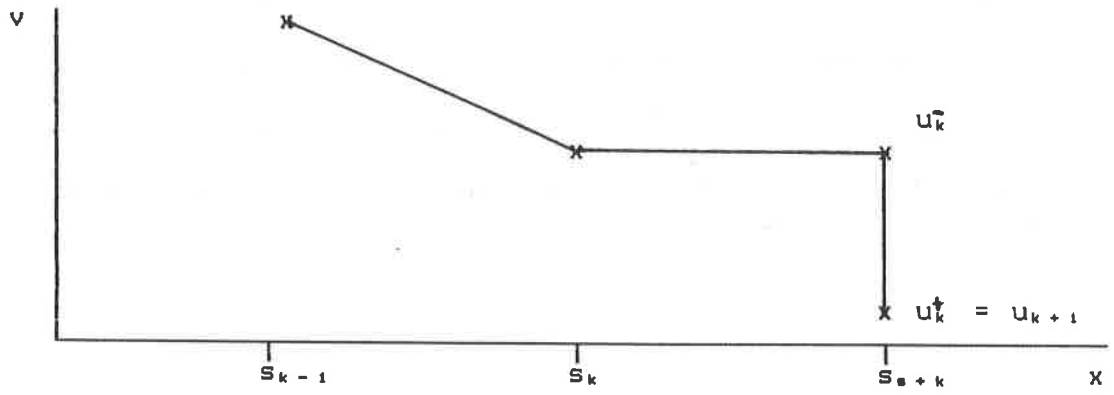
This problem is overcome by using the the NR algorithm (implicit or explicit version) with step 5 amended as below.

5a) Assume at time level n the shock is at node k, having values of u_k^- & u_k^+.(see Fig.4.33)

5b) Take node k to be single valued with value u_k^- and node k+1 to be double valued with upper value u_k^- and lower value $u_k^+ = u_{k+1}$.(see Fig.4.34)

5c) Advance the solution one time step, with the speeds of the nodes on grid M_t set to $\dot{s}_n$ (see steps 3 & 4 of NR algorithm)

Fig.4.34 Diagram showing double valued point at s_{k+1}. i.e end of step 5b.



4.3.2 Shock Test Problems.

It was decided to apply the above algorithm and the ND algorithm to two simple test problems. First we consider the linear advection equation

$$u_t + u_x = 0 \qquad (4.22)$$

where the initial data is chosen as the unit step function

situated at x = 0.5., the speed of the front being calculated from the Rankine-Hugoniot shock condition, which gives

$$\dot{s}_* = 1 \tag{4.23}$$

The grids are set up as in section 4.2.1, with the shock centred at x= 0.5 on grid M_t, the speed of the shock (nodes on grid M_t) being given by equation 4.23 above. As $< u_x , \alpha_i > = 0$ for all i (assuming an internal boundary at the shock (see Wathen[4])), then $u_t = 0$ for both algorithms up to the point of node reversal/deletion. During reversal (NR) node k+1 is double valued using the routine above, which causes $m_k = 0$ therefore preserving the exact solution through node reversal (see Fig.4.35). Also the ND algorithm gives the exact solution during deletion, since the solution is a constant to the right of the shock (see Fig.4.36).

The second test problem is the inviscid Burgers' equation

$$u_t + uu_x = 0 \tag{4.24}$$

using the same initial data and grids as for the linear advection equation, with the speed of the shock in this case being given by

$$\dot{s}_* = \tfrac{1}{2}(u_{k+1} - u_k) \tag{4.25}$$

Again both methods exhibit the true solution (Figs.4.37 & 4.38) by the same argument as above.

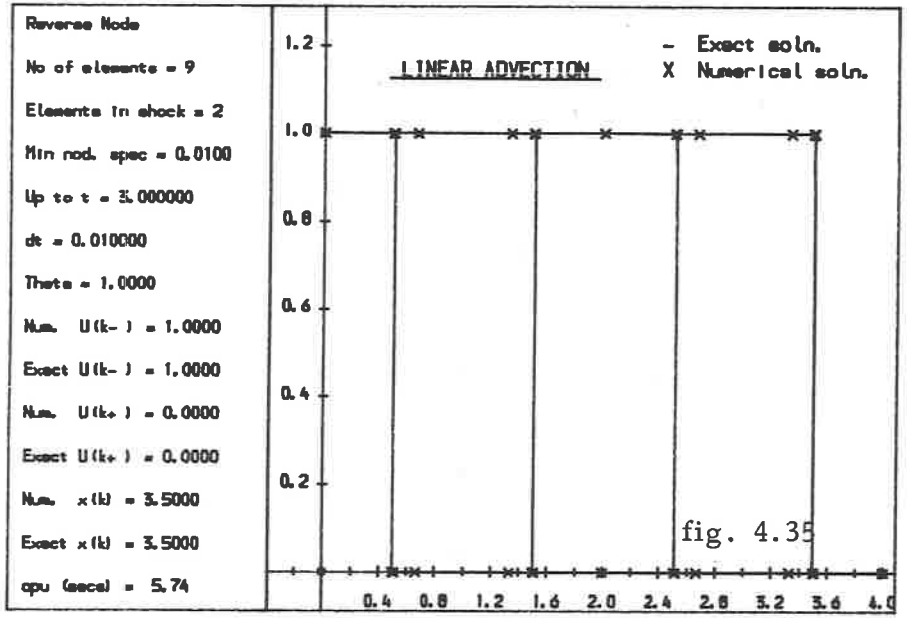


fig. 4.35

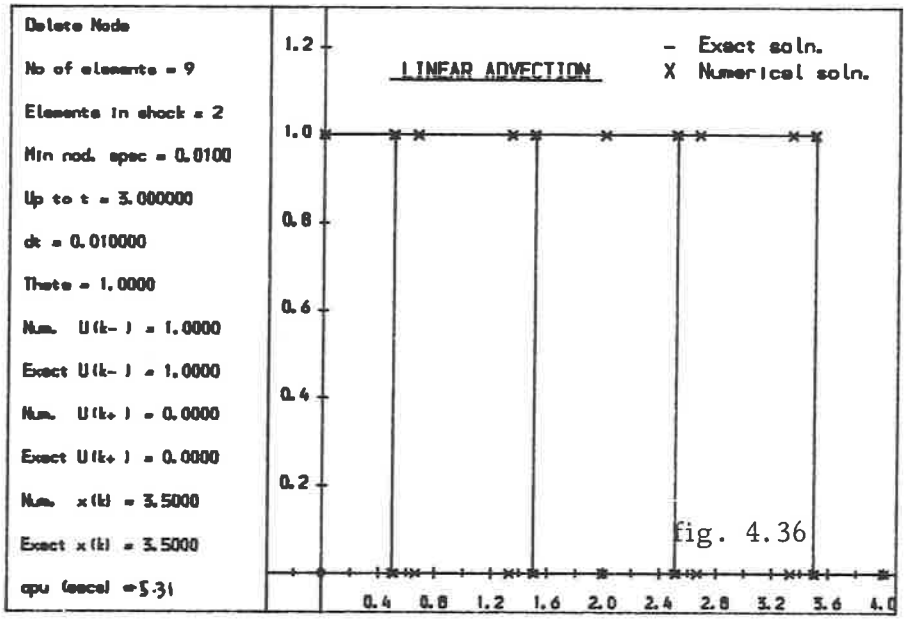


fig. 4.36

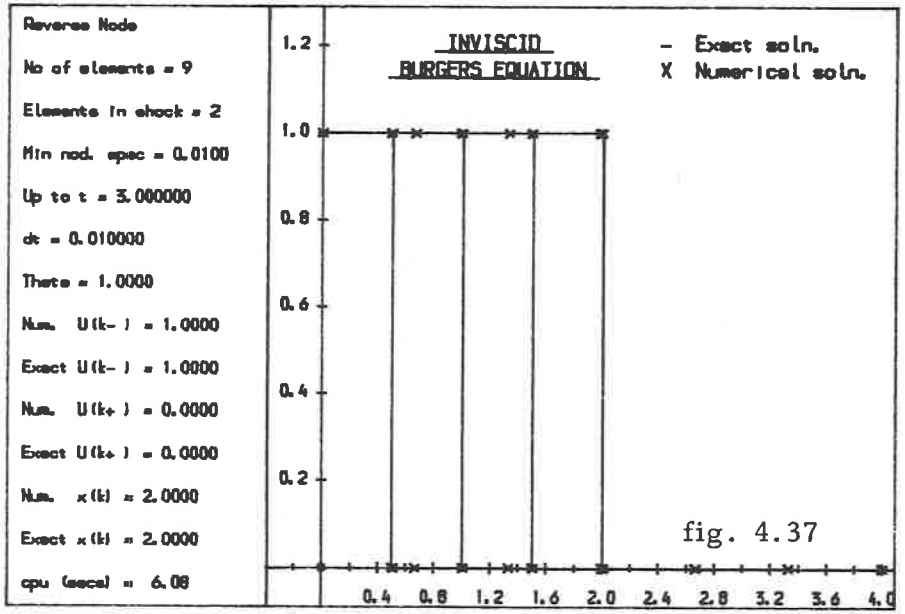


fig. 4.37

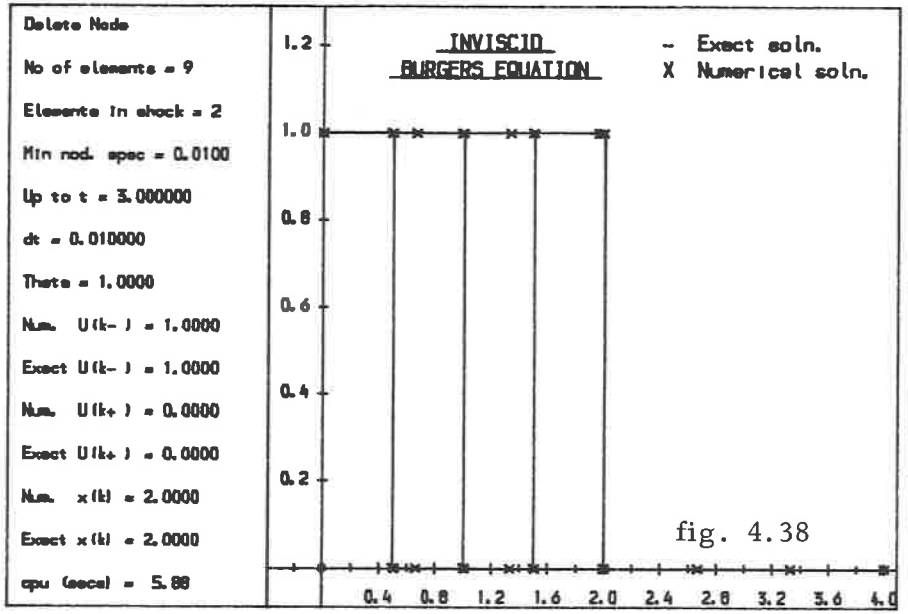


fig. 4.38

4.3.3 The Buckley-Leverett Equation.

The final test problem considered is again related to oil recovery (see e.g. Wathen [4]), i.e. the Buckley-Leverett equation which may be used to model the displacement of oil by water in a water driven recovery process. First we consider the inviscid (or homogeneous) form of the equation (assuming no capillary pressure) in which a shock is present, and finally the viscous form of the equation in which a steep front is present.

The inviscid form of the equation can be written as

$$u_t + f(u)_x = 0 \tag{4.26}$$

where

$$f(u) = \frac{u^2}{u^2 + \frac{1}{2}(1-u)^2} \tag{4.27}$$

The boundary condition on the upstream side (x=0) is taken to be u(0) = 1 with the initial data chosen such that a shock is present at x = 1.3 with $u_k^- = 0.4$, $u_k^+ = 0.2$, with a tan curve fitted to the left of the shock and a quadratic to the right. Using the method of characteristics it is possible to obtain an exact solution to this problem, and a comparison between this and the numerical solution can be seen in Fig.4.39 for the NR algorithm and in Fig.4.40 for the ND algorithm, where the initial grids are set up as in the linear advection case with the speed of the shock given by

$$\dot{s}_k = \frac{[f(u_k^-) - f(u_k^+)]}{u_k^- - u_k^+} \tag{4.28}$$

Using the same equations it was then decided to use some more

realistic initial data (see Wathen [12]), the shock initially being at x = 0.5, with the value to the right being zero. The results in this case for both methods are shown in figs.4.41 & 4.42, with very good comparisons between the numerical and exact value at the top of the shock at time t = 1.5 (within 0.6%).

Finally equation (4.26) was modified to include a viscous term (see [13]), i.e.

$$u_t + f(u)_x = \varepsilon u_{xx} \qquad (4.29)$$

where ε was chosen to be 10^{-3} , the initial data being such that instead of having a shock at x = 1.3, a steep front is centred here, and thus the initial grids are distributed as in section 4.2.1, First runs were done using a quadratic to the left of the front (see Figs.4.43 & 4.44). Although an exact solution does not exist to this problem the results compared favourably with those for the inviscid case. Finally the initial data was chosen to be zero to the left of the front, with results which can be seen in Figs.4.45 and 4.46. Here it should be noted that the solution at the top of the front exhibits considerably more curvature than in the inviscid case, probably due to the fact that the second derivative u_{xx} exists at this point due to the nature of the viscous equation, where this is not the case for the inviscid equation.

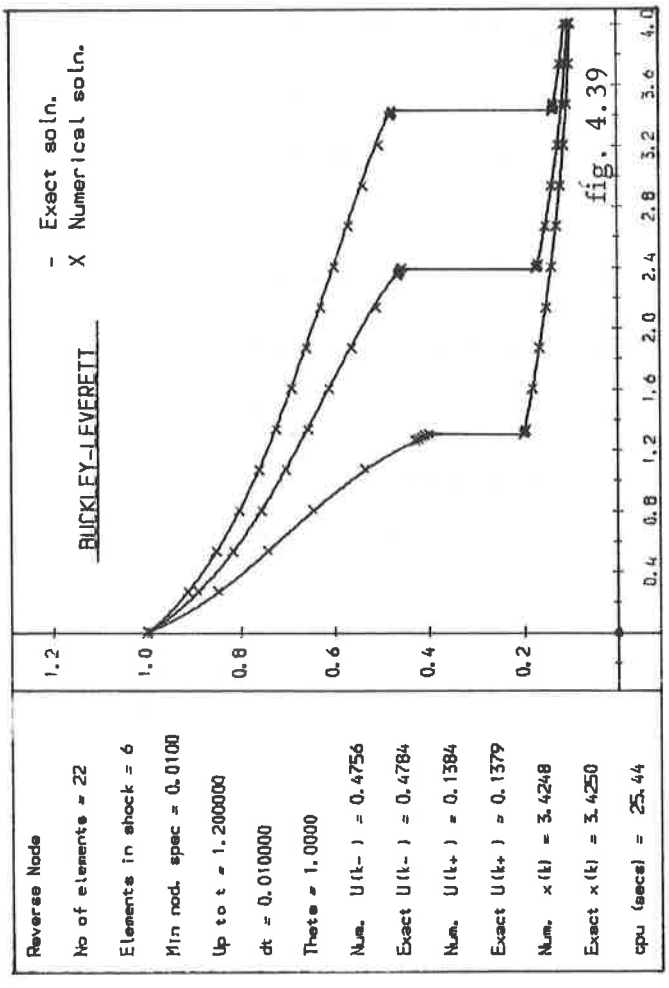


fig. 4.39

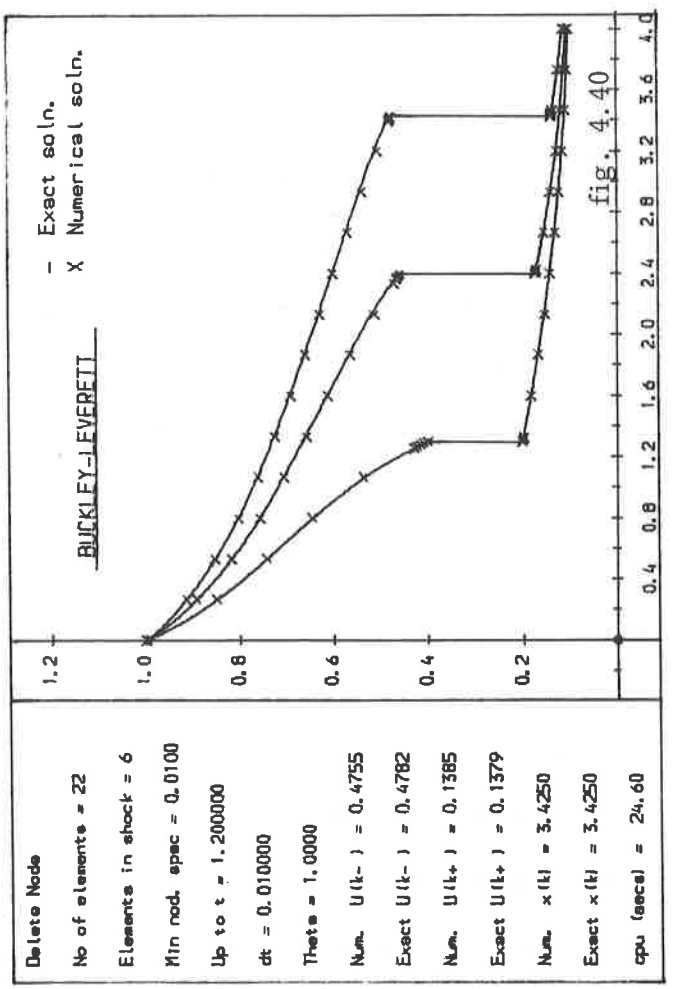


fig. 4.40

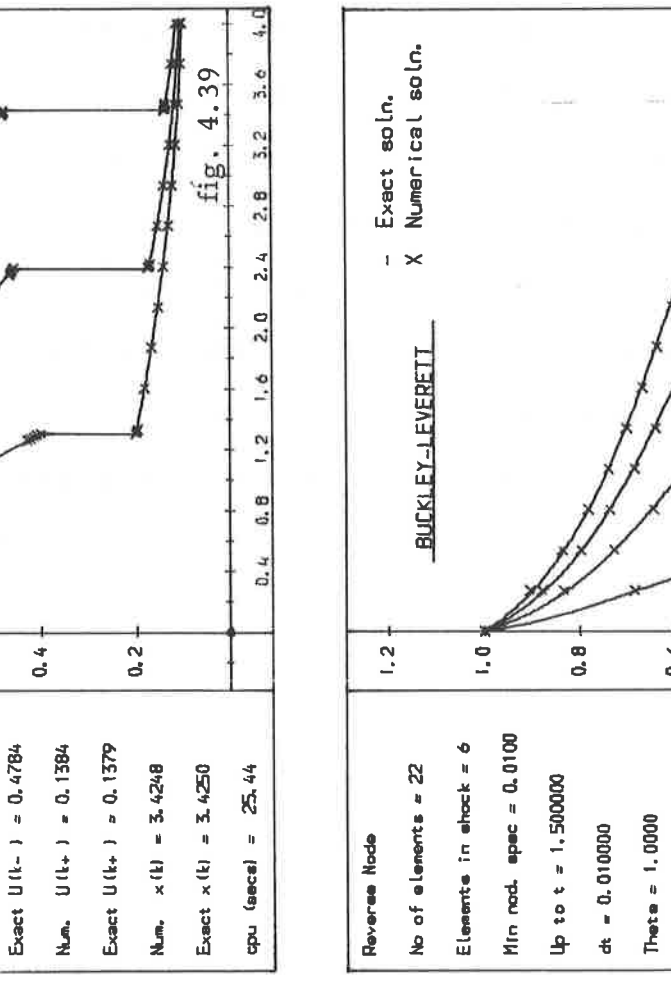


fig. 4.41

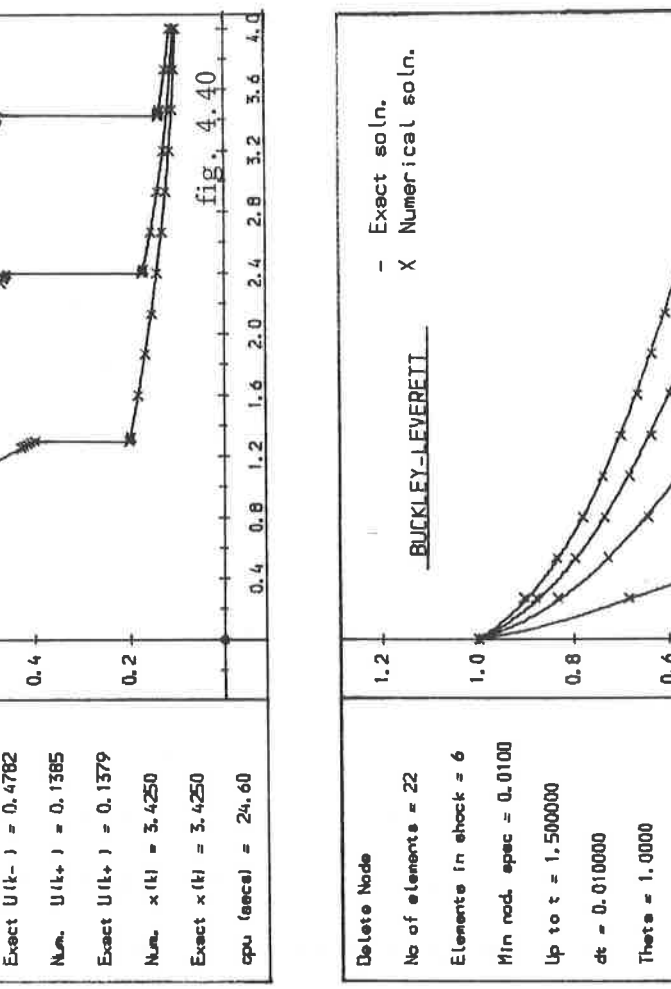


fig. 4.42

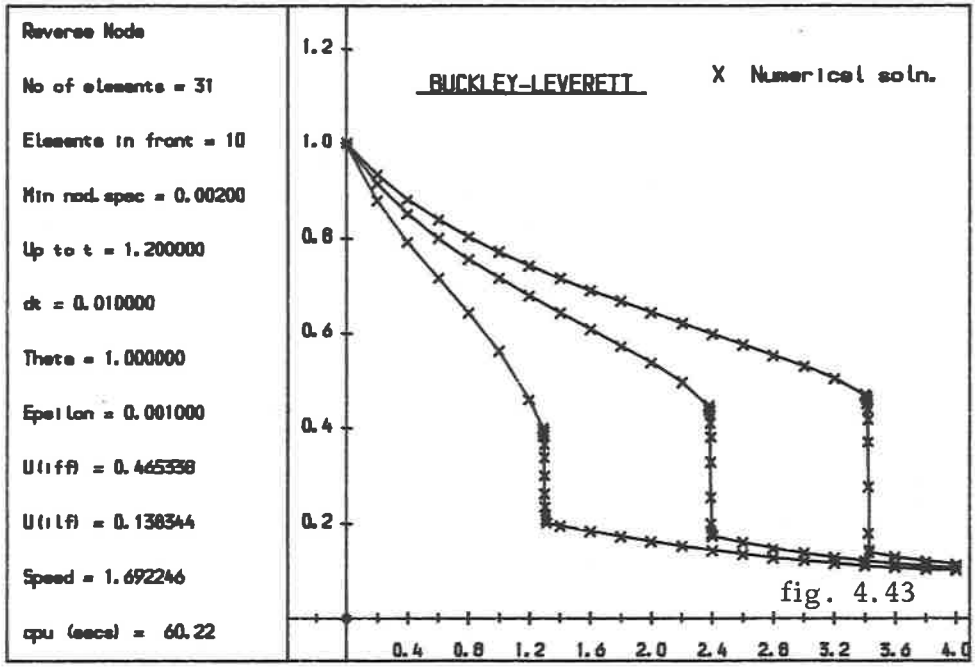


fig. 4.43

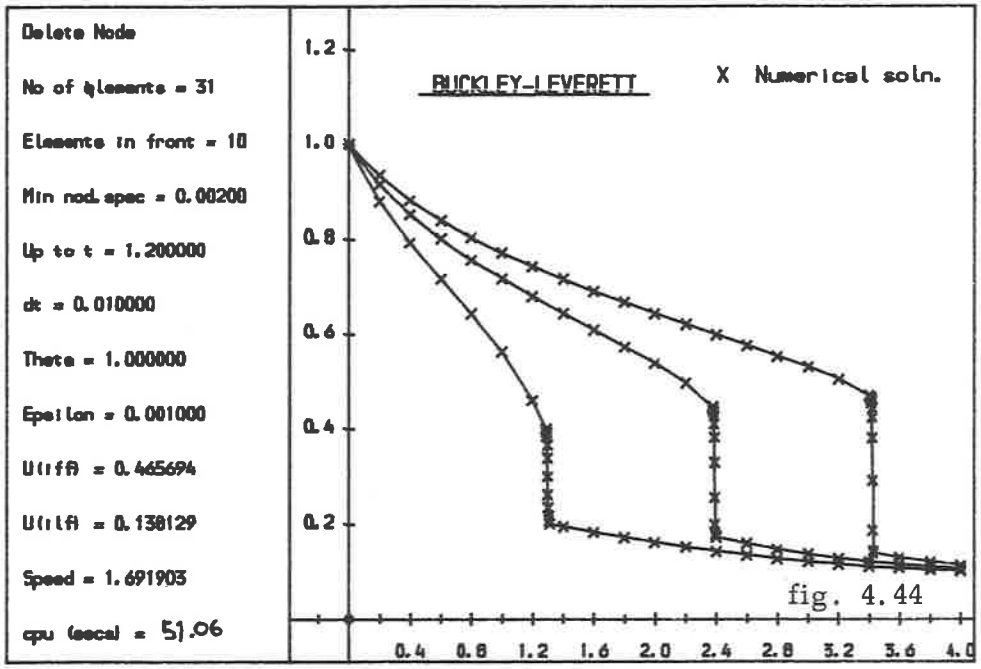


fig. 4.44

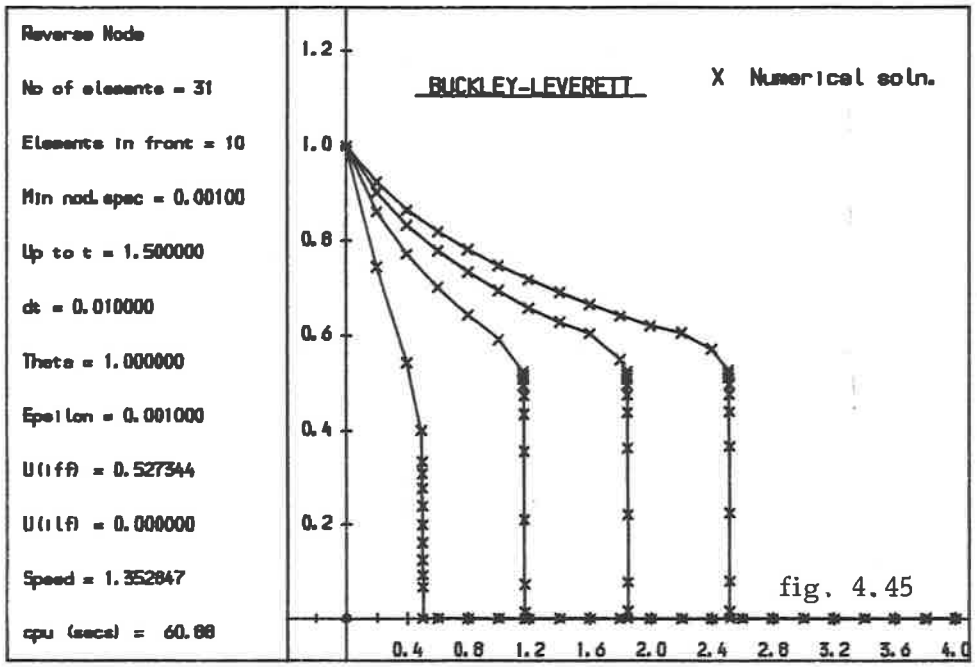


fig. 4.45

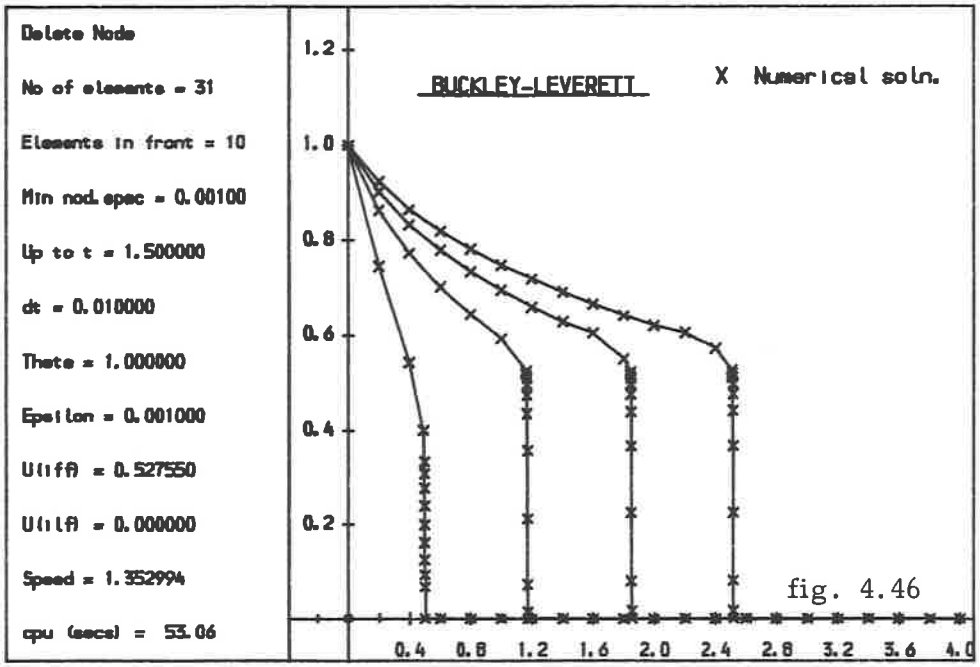


fig. 4.46

5. Conclusions

In this report we have developed two regridding techniques, based on an underlying fixed grid F_o with a moving grid M_t superimposed on it, the motion of the nodes on the moving grid being governed by a constrained MFE method. In section 4.1 the method was applied to a moving boundary problem, where it was found possible to implement an implicit algorithm (section 4.1.5 & 4.1.6) which gave very good results with a vast decrease in cpu time over the corresponding explicit method. In section 4.2 the method was tested on Burgers' equation, for a solution where the tracking of a steep front was important, and here again good results were obtained. Finally in section 4.3 the method was adapted to cope with shock problems, the Buckley-Leverett results showing that the algorithms were successful.

Using an underlying fixed grid gives one the added advantage that if a system of equations is to be solved the same grid can be used on all the equations. If only one of the components develops a shock, then the methods can track the shock and also maintain a fixed grid for use with the other components. If an MFE method were used the grid would have to be generated from the equations which contain the shocked components and the grid may not be suited to the solution of other equations of the system.

It is hoped to next apply the algorithms described here to a system of equations that model steam injection, where it is important to track the boundary between the steam and the

water (which may be in the form of a steep front or shock) and also maintain a fine grid in certain areas of the solution domain in order to capture particular physical effects (i.e. pressure drop at the well bore). Finally it is hoped in the future to extend the methods mentioned in this report into 2-d.

Acknowledgements

It is with pleasure that I acknowledge the help and encouragement given to me by Dr. M. J. Baines throughout this work. I would also like to thank R. O. Moody for several very interesting discussions, and Dr C. P. Please for some useful suggestions concerning this work.

Finally I wish to thank M. H. Goldwater at BP Petroleum Development, for his ideas and interest over the last two years.

I acknowledge the support of the S.E.R.C. through a CASE studentship with BP.

Appendix

We show here that if an implicit time-stepping method is used to track a moving boundary, then after n time steps the numerical value obtained will overestimate the true location.

Using the moving boundary problem of section 4.1.4, where the true boundary velocity is given by equation (4.6)

$$s(t) = s_0 e^{t/2} \qquad (4.6)$$

the speed is estimated from equation (4.3), which can be written as

$$s(t) = \frac{k_L u_x^- - k_R u_x^+}{4(k_L - k_R)} \quad , \qquad (A.1)$$

the exact solution being given by equation (4.6), namely

$$u(x,t) = x^2 - s_0^2 e^t \qquad (4.6)$$

$$u_x = 2x \qquad x \in [a,b]$$

If we assume that the numerical solution correctly estimates the slope either side the boundary then from equation (A.1) we can write the speed of the boundary, noting that the slope at the boundary is 2s, as

$$s(t) = \frac{s(t)}{2} \qquad (A.2)$$

If we approximate s(t) with

$$s^{n+1} = \frac{s^{n+1} - s^n}{dt} = \frac{s^{n+1}}{2} \qquad (A.3)$$

$$s^{n+1} = \frac{s^n}{(1 - dt/2)} \qquad (A.4)$$

then it can also be shown for the exact solution (4.6) that

$$s^{n+1} = s^n e^{dt/2} \qquad (A.5)$$

and, since

$$(1 - x)^{-1} > e^x \quad 1 > x \geq 0 \quad , \qquad (A.6)$$

it can be seen that for $dt < 2$ the numerical estimation of the boundary position (A.4) is greater than the exact position.

References

[1] WATHEN, A.J. & BAINES, M.J. (1985), IMA J. Numer. Anal. 5, 161-82.

[2] MOODY, R.O. (1985), Numer. Anal. Report 17/85, Dept. of Maths., Univ. of Reading.

[3] MOODY, R.O. (1987), Numer. Anal. Report 10/87, Dept. of Maths., Univ. of Reading.

[4] WATHEN, A.J. (1984), Ph.D. Thesis, Univ. of Reading.

[5] MILLER, K. & MILLER, R. (1981), SIAM J. Numer. Anal. 18(6). 1019-1057.

[6] LYNCH, D.R. (1982), J. Comput. Phys. 47, 387-411.

[7] MUELLER, A. (1983), Ph.D. Thesis, Univ. of Texas at Austin.

[8] COUCH, E.J. KELLER, E.J. WATTS, & J.W. (1970), J. Can. Pet. Tech. April-June 1970, 107-111.

[9] CIAVALDINI, F.L. (1970), Int. Chem. Engng. 10(1), 42-8.

[10] HEATH, D.E. (1986), Numer. Anal. Report 17/86, Dept. of Maths., Univ. of Reading.

[11] JOHNSON, I.W. (1986), Ph.D. Thesis, Univ. of Reading.

[12] WATHEN, A.J. (1982), Numer. Anal. Report 4/82, Dept. of Maths., Univ. of Reading.

[13] GELINAS, R.J. DOSS, S.K. & MILLER, K. (1981), J. Comput. Phys. 40, 202-249.