10
10
Φ8cmcsc8

TEMPORAL ERROR CONTROL FOR CONVECTION-DOMINATED EQUATIONS IN TWO SPACE DIMENSIONS.

M. BERZINS*

Abstract. A new time integration strategy for the solution of convection-dominated partial differential equations in two space dimensions by the method of lines is presented. The strategy aims to ensure that the time integration error is less than the spatial discretisation error. This is achieved by making use of the individual contributions of the local spatial discretisation error and the local time integration error to the global error in the numerical solution. Numerical results are used to illustrate the performance of this strategy.

Key words. Convection-dominated equations, method of lines, spatial and temporal errors.

AMS subject classifications. 65M20, 65M15

1. Introduction. Much progress has been made recently in the spatial discretisation of convection-dominated equations. The finite volume spatial discretization methods now used provide numerical solutions that are free of spurious oscillations. Such discretizations are considered by many authors e.g. by Spekreijse [16] and Koren [10] for the steady Euler (and also Navier-Stokes) equations using quadrilateral meshes. Other authors use triangular meshes, for example Ware et. al. [18] and Durlofsky et. al. [5]. Time-dependent problems may be solved by discretising in space and using o.d.e. initial value problem software to integrate forward in time using a method of lines type approach.

The spatial discretisation error introduced by the semi-discretization may be controlled by the use of one of the many adaptive algorithms available, e.g. see Strouboulis and Oden [17], Berger and Oliger [1] or Moore and Flaherty [14]. Such algorithms will automatically refine and coarsen the spatial mesh as part of a spatial error control procedure. When using such techniques as part of solving a time dependent problem it is still necessary to integrate in time with sufficient accuracy so that the spatial error is not degraded while maintaining efficiency. This is the subject of this paper and is achieved by calculating the time accuracy tolerance automatically using estimates of the spatial error. The starting points for this work are the error balancing approaches of Lawson, Berzins and Dew [11] for parabolic equations and the demonstration of a different approach for one space dimensional hyperbolic equations by Berzins [2], who also discusses earlier related work by Berger and Oliger [1] and Gary [6]. The aim in this paper is to show that the new approach works for two-dimensional convection dominated problems and to provide a justification for the approach used here and in [2].

This new approach is in contrast to most existing methods for the time integration of p.d.e.s in which one of two approaches is taken. Either a stability control and a specialised time integration method is employed, e.g. see [9], or an an o.d.e. solver is used which makes use of local time error per step, [6], control. In the first case the solution may be stable but not necessarily accurate, while in the second case the user-supplied accuracy tolerance may be difficult to relate to the spatial discretisation error and also to the global time error in the computed solution.

The class of problems considered in this paper is given in Section 2 together with the spatial discretisation method used. Section 3 describes the time integration algo-

* School of Computer Studies, The University, Leeds LS2 9JT.

rithm. Section 4 outlines the background theory behind the error balancing algorithm and describes the new error control strategy. The analysis of the new strategy shows that the time integration error remains below the spatial discretisation error and that the strategy is a form of local error per unit time step control. Section 5 describes the implementation of the new method and shows that there is an implicit CFL condition in the new approach. Numerical experiments illustrating the performance of the new method are described in Section 6 and a summary presented in Section 7. It is shown that the error control strategy appears to offer an effective means of ensuring that the spatial error dominates the temporal errors in the method of lines solution of convection-dominated equations.

The eventual goal of this research is to devise reliable automatic algorithms for convection-dominated p.d.e.s, similar to algorithm for parabolic p.d.e.s of Lawson and Berzins [12] in which the user provides only a specification of the problem, an initial spatial mesh and an error tolerance for the overall error. A prototype algorithm of this type for two-dimensional hyperbolic equations is described by Berzins, Lawson and Ware [4]. The algorithms are intended to be *reliable* in that they make use of spatial and temporal error estimates to meet automatically the users accuracy requirements.

2. Problem Class and Spatial Discretisation. The solution strategy for time-dependent p.d.e.s is to follow Koren [10] by splitting the equations into their convective and diffusive parts. This enables upwind discretization methods developed for the Euler equations to be used for the convective part of the system and the centered discretization methods to be used for the diffusive part. The class of p.d.e.s to be considered is written in cartesian co-ordinates as

$$\frac{\partial q}{\partial t} + \frac{\partial}{\partial x}(f_1(q) + f_2(q)) + \frac{\partial}{\partial y}(g_1(q) + g_2(q)) = 0, \ t \ \epsilon \ (0, t_e], \ (x, y) \ \epsilon \ \Omega, \tag{1}$$

with appropriate boundary and initial conditions. In the case of the compressible Navier Stokes equations for instance the solution vector has the form $q(x, y, t) = [e, \rho, \rho u, \rho v]^T$. Here, ρ is the fluid density; u, v are the cartesian components of the velocity vector, e is the internal energy. The pressure p is evaluated according to an equation of state. The fluxes f_1 and g_1 represent the convective fluxes while f_2 and g_2 are the diffusive fluxes, [10]. Although the ideas developed in this paper are applicable to equations of this form, and indeed have been applied to such equations, see Ware and Berzins [18], for ease of exposition a single equation of this type will be considered:

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(f(u)) + \frac{\partial}{\partial y}(g(u)) = \nu(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) \ , \ t \ \epsilon \ (0, t_e], \ (x, y) \ \epsilon \ \Omega. \tag{2}$$

In this study it will be assumed that convection is the dominant mechanism and so that ν is small. For simplicity the boundary and initial conditions are taken to be of the form

$$u(x, y, t) = g(x, y, t) \ , \ (x, y) \ \epsilon \ \delta\Omega \ , \ t \ \epsilon \ (0, t_e]. \tag{3}$$

$$u(x, y, 0) = u_0(x, y) \ , \ (x, y) \ \epsilon \ \Omega. \tag{4}$$

We assume that the p.d.e. defined by the above equation is well-posed and has a unique solution.

The numerical solution method used to solve (2), (3) and (4) is the method of lines in which the p.d.e. is discretised in space and the resulting o.d.e. system in time is solved using o.d.e. software. Although discretization methods of the type used here have been applied to general quadrilateral meshes, [10], and to unstructured triangular meshes [18], for the sake of simplicity a regular square mesh will be considered. A mesh on the region $\Omega = [0,1]x[0,1]$ is defined by the points (x_i, y_j) where

$$x_i = (i - \frac{1}{2})\,\Delta x \quad \text{and} \quad \mathrm{y_j} = (\mathrm{j} - \frac{1}{2})\,\Delta \mathrm{y} \quad \mathrm{i} = 1, ..., \mathrm{N}, \mathrm{j} = 1, ..., \mathrm{N}\ ,$$

and where $\Delta x = \Delta y = 1/(N+1)$.
A finite volume approach is used and equation (2) is integrated over a square cell whose centre is at the point (x_j, y_k) . Denote by $U_{j,k}(t)$ the numerical solution at the point (x_j, y_k) . Applying the divergence theorem with one point quadrature rule used to integrate along the edges of the cell (see [10]) to obtain

$$\begin{aligned} \Delta x \Delta y \frac{dU_{j,k}(t)}{dt} &+ [f_{j+\frac{1}{2},k} - \nu(U_{j+\frac{1}{2},k})_x - f_{j-\frac{1}{2},k} + \nu\ (U_{j-\frac{1}{2},k})_x]\Delta y \\ &+ [g_{j,k+\frac{1}{2}} - \nu(U_{j,k+\frac{1}{2}})_y - g_{j,k-\frac{1}{2}} + \nu(U_{j,k-\frac{1}{2}})_y\ ]\Delta x = 0. \end{aligned} \tag{5}$$

where $(U_{j+\frac{1}{2},k})_x$ is the space derivative of U w.r.t. x at the point $(x_{j+\frac{1}{2}}, y_k)$ and where

$$x_{j+\frac{1}{2}} = \frac{x_j + x_{j+1}}{2},\ x_{j-\frac{1}{2}} = \frac{x_j + x_{j-1}}{2},\ y_{k+\frac{1}{2}} = \frac{y_k + y_{k+1}}{2},\ y_{k-\frac{1}{2}} = \frac{y_k + y_{k-1}}{2},$$

$$j = 2, ..., N-1 \text{ and } \mathrm{x}_{\frac{1}{2}} = \mathrm{y}_{\frac{1}{2}} = 0\ ,\ \mathrm{x}_{\mathrm{N}+\frac{1}{2}} = \mathrm{y}_{\mathrm{N}+\frac{1}{2}} = 1.$$

As the solution values are only piecewise constant inside each cell the evaluation of the convective fluxes midway along the edge involves the approximate solution of four one-dimensional Riemann problems in the direction of the normals to the edges of the quadrilateral. This is done by using the upwind scheme of Osher and Engquist as described by Koren [10], who shows that this is an efficient, accurate, and robust means of treating convection problems. The convective fluxes $f_{j+\frac{1}{2},k}$ and $f_{j-\frac{1}{2},k}$ are defined by

$$\begin{aligned} f_{j+\frac{1}{2},k} &= \hat{f}(x_{j+\frac{1}{2}}, y_k, t\ , U^x_{j,k}(x_{j+\frac{1}{2}}\ ,\ y_k,\ t)\ ,\ U^x_{j+1,k}(x_{j+\frac{1}{2}}\ ,\ y_k,\ t)\ ) \\ f_{j-\frac{1}{2},k} &= \hat{f}(x_{j-\frac{1}{2}}, y_k, t\ , U^x_{j-1,k}(x_{j-\frac{1}{2}}\ , y_k,\ t)\ ,\ U^x_{j,k}(x_{j-\frac{1}{2}}\ , y_k,\ t)\ ) \end{aligned}$$

where $U^x_{j,k}(x, y, t)$ is the x dimensional upwind interpolant from cell j, k evaluated at the mid-point (x, y, t) of an edge. The flux function $\hat{f}$ is defined by the solution of a Riemann problem with $U^x_{j,k}(x, y, t)$ and $U^x_{j+1,k}(x, y, t)$ as the discontinuous values on each side of the edge. These values are calculated by x-dimensional upwind interpolants from cells j, k and $j+1, k$ respectively. The convective fluxes $g_{j,k+\frac{1}{2}}$ and $g_{j,k-\frac{1}{2}}$ are defined in the same way by

$$\begin{aligned} g_{j,k+\frac{1}{2}} &= \hat{g}(x_j, y_{k+\frac{1}{2}}, t\ , U^y_{j,k}(x_j, y_{k+\frac{1}{2}},\ t)\ ,\ U^y_{j,k+1}(x_j, y_{k+\frac{1}{2}},\ t)\ ) \\ g_{j,k-\frac{1}{2}} &= \hat{g}(x_j, y_{k-\frac{1}{2}}, t\ , U^y_{j,k-1}(x_j, y_{k-\frac{1}{2}},\ t)\ ,\ U^y_{j,k}(x_j, y_{k-\frac{1}{2}}\ ,\ t)\ ) \end{aligned}$$

where $U^y_{j,k}(x, y, t)$ is the y dimensional upwind interpolant from cell j, k evaluated at the mid-point (x, y, t) of an edge. The function $\hat{g}$ is defined by the solution of a

Riemann problem with $U^y_{j,k}(x,y,t)$ and $U^y_{j,k+1}(x,y,t)$ as the discontinuous values on each side of the edge.

Although the piecewise constant values on either side may be used this only gives a first-order scheme. Koren [10], for example, shows how upwind left and right piecewise linear interpolants can be built up on the edge by using solution values from the quadrilaterals and their neighbours on either side. Limited combinations of these interpolants can be used to produce more accurate estimates of solution values on the edge. For instance

$$U^x_{j-1,k}(x_{j-\frac{1}{2}},y_k,t) = U_{j-1,k}(t) + \frac{\Delta x}{2}\ \frac{(U_{j-1,k}(t)-U_{j-2,k}(t))}{\Delta x} B(r^x_{j-1},1)$$
$$\text{and}\quad \mathrm{U^x_{j,k}(x_{j-\frac{1}{2}},y_k,t) = U_{j,k}(t) - \frac{\Delta x}{2}\ \frac{(U_{j+1,k}(t)-U_{j,k}(t))}{\Delta x} B(\frac{1}{r^x_j},1)},$$

where the ratio r^x_j is defined by

$$r^x_j \ = \ \frac{U_{j+1,k}(t)-U_{j,k}(t)}{U_{j,k}(t)-U_{j-1,k}(t)}\ . \tag{6}$$

Roe, [15], shows that, on uniform meshes, different limiter functions, $B(.,.)$, give rise to different spatial accuracies. Three useful choices of limiter are:
First Order Method:

$$B_1(r^x_j,1) \ = \ 0 \tag{7}$$

Second Order Method: (reverts to first-order if $r^x_j \le 0$)

$$B_2(r^x_j,1) \ = \ \frac{r^x_j + |r^x_j|}{1+|r^x_j|} \tag{8}$$

Third Order Method:

$$B_3(r^x_j,1) \ = \ 0.25 + 0.75 r^x_j \tag{9}$$

where in each case the ratio r^x_j is defined as in equation (6) above. The first order scheme smears shocks excessively. The second order scheme is due to Van Leer while the third order scheme is due to Leonard and produces non-monotone solutions, see Roe [15]. For these reasons the first and third-order schemes will be used mainly as part of the spatial error estimation process. The third-order scheme may be modified to produce monotone solutions, as was done by Gaskell and Lau [7] (in somewhat different notation) to get

$$B_4(r^x_j,1) \ = \ \max(0,\min(2\mathrm{r^x_j},\min(0.25+0.75\mathrm{r^x_j},4))).$$

The boundary conditions are, for the sake of simplicity, assumed to define the solution on the boundary as in equation (3). These exact boundary solution values are used in the appropriate flux and derivative calculations. For purely hyperbolic equations the method of characteristics is used to determine the boundary conditions. The initial condition is defined by evaluating the function $u_0(x,y)$ at the spatial mesh points

$$U_{i,j}(0) \ = \ u_0(x_i,y_j) \ \ , \ i=1,...,N \ \ , \ j=1,...,N.$$

The system of differential equations in time (5), after dividing by $\Delta x \Delta y$, may be written in vector form as:

$$\dot{\underline{U}} \ = \ \underline{F}_M \ \ (\ t, \ \underline{U}(t) \ ), \tag{10}$$

where the M dimensional vector , $M = N \ \times \ N \quad$, is defined by

$$[\underline{U}(t)]_k \ = \ U_{i,j}(t) \quad k \ = \ (i-1) \ \times \ N + j, \ \ i = 1, ..., N \ \ , \ j = 1, ..., N. \tag{11}$$

In the cases considered here the system of differential equations (10) is non-stiff as the p.d.e. (1) has convective terms dominating small diffusive terms.

3. Time Integration of Conservation Laws. The error in the numerical solution depends on both the spatial discretisation error and on the time integration error. The approach adopted here is to decouple these two errors and to try and control them individually. The initial value problem to be solved is given by equation (10) with the true solution $[\underline{U}(t_n)]_{n=0}^p$ approximated by $[\underline{V}(t_n)]_{n=0}^p$ at a set of discrete times $0 = t_0 < t_1 < ... < t_p = t_e$ by a time integration method. The vector of the values of the overall error at the spatial mesh points, at any time t, is defined by $\underline{E}(t)$ where

$$\underline{E}(t) \ = \ \underline{u}(t) - \underline{V}(t) \tag{12}$$

and where $\underline{u}(t)$ is the restriction of the exact p.d.e. solution to the mesh i.e.

$$[\ \underline{u}(t) \ ]_k \ = \ u(x_i, y_j, t) \ , \ k \ = \ (i-1) \times N + j, \ i = 1, ..., N \ \ , \ j = 1, ..., N.$$

The vector $\underline{E}(t)$ may also be written as a combination of the restriction of the p.d.e. spatial discretisation error $\underline{e}(t)$, which represents the spatial discretisation error at the mesh points and is defined by

$$\underline{e}(t) \ = \ \underline{u}(t) \ - \ \underline{U}(t), \tag{13}$$

and the o.d.e. global error $\underline{g}(t)$; that is,

$$\begin{aligned} \underline{E}(t) = \underline{u}(t) - \underline{V}(t) \ &= \ (\underline{u}(t) - \underline{U}(t)) + (\underline{U}(t) - \underline{V}(t)) \\ &= \ \ \underline{e}(t) \ + \ \underline{g}(t). \end{aligned} \tag{14}$$

In order that the time integrator is used efficiently, the temporal error $\underline{g}(t)$ should not dominate the spatial discretisation error $\underline{e}(t)$ but nor should the o.d.e.s be integrated with a much higher degree of accuracy than that already attained in space. It follows that the spatial and temporal errors should be roughly the same order of magnitude, although in practice the spatial discretisation error must dominate. This dominating error (and hence the overall error) can then be controlled by spatial remeshing. As the spatial error varies during the integration it is difficult to select a single o.d.e. tolerance that will ensure that the spatial error dominates. Thus the time accuracy tolerance must be proportional to the spatial discretisation error.

Most of the codes available for solving time-dependent o.d.e.s, attempt to control the local time integration error in the computed solution with regard to a user-supplied accuracy tolerance, *tol*. The local error is defined in terms of the local solution on $[t_n, t_{n+1}]$, $\underline{y}_{n+1}(t)$, which is the solution of the o.d.e.

$$\dot{\underline{y}}_{n+1}(t) = \underline{F}_M(t, \underline{y}_{n+1}(t)), \ \underline{y}_{n+1}(t_n) = \underline{V}(t_n). \tag{15}$$

The local error per step (LEPS) at t_{n+1} is then given by

$$\underline{le}_{n+1}(t) = \underline{V}(t_{n+1}) - \underline{y}_{n+1}(t_{n+1}) \ . \tag{16}$$

The local error per unit step (LEPUS) is given by $\underline{le}_{n+1}(t)/k_n$ where $k_n = t_{n+1} - t_n$.

3.1. The Adaptive Theta Method of Berzins and Furzeland. The Theta (θ) method algorithm of Berzins and Furzeland, [3], can handle both stiff (diffusion or source term dominated) and non-stiff (convection-dominated) problems by automatically selecting the best value of theta and the most efficient iteration method (Newton or functional iteration) with respect to step size and cost per step. The criteria for selecting theta and for switching are established by optimising the permissible step size.

Berzins [2] shows that the method has similarities with the iterated Leap Frog method of Hyman [9] in that Hyman's method may be viewed as the θ method with local extrapolation and a value of θ which depends on the present and previous stepsizes.

The Theta method is defined as follows. Given the approximation $\underline{V}(t_n)$ to $\underline{U}(t_n)$, and the approximation $\dot{\underline{V}}(t_n)$ to $\dot{\underline{U}}(t_n)$, then the numerical solution at t_{n+1} , where $t_{n+1} = t_n + k$ and k is the time step size, is denoted by $\underline{V}(t_{n+1})$ and is defined by

$$\underline{V}(t_{n+1}) = \underline{V}(t_n) + (1-\theta)k\dot{\underline{V}}(t_n) + \theta k\underline{F}_M(t_{n+1}, \underline{V}(t_{n+1})), 0 < \theta < 1. \tag{17}$$

As this formula is implicit in $\underline{V}(t_{n+1})$, an iterative method and an initial predictor for $\underline{V}^{(0)}(t_{n+1})$, see [3], are needed. The simplest iterative method generates successive correctors from the functional iteration formula:

$$\underline{V}^{(m+1)}(t_{n+1}) = \underline{V}(t_n) + (1-\theta)k\ \dot{\underline{V}}(t_n) + k\theta\ \underline{F}_M(t_{n+1}, \underline{V}^{(m)}(t_{n+1})), \tag{18}$$

where $m = 0, 1, ..$.
The condition for the convergence of functional iteration with a rate of convergence r_c is

$$k\ L\ \theta \ < \ r_c \quad \text{where } \mathrm{r_c} \ < \ 1 \ , \tag{19}$$

$L = \ Sup \ || \ J \ ||$, J is the Jacobian matrix $\left[\frac{\partial \underline{F}_M}{\partial \underline{V}_{n+1})}\right]$ evaluated at t and an unknown point, $\underline{V}(t_{n+1})$, and where L is a suitable Lipschitz constant for some matrix norm. The Theta code used here estimates L as in [3] by using the computed rate of convergence of functional iteration. A value of $r_c = 0.3$ is used and the iteration is terminated when convergence is sufficiently fast and the final correction to the solution is sufficiently small.

An alternative approach is to use the iteration (18) for a fixed number of times without insisting that (19) be used. This is similar to the approach used in some Adams method codes and means that only the local error test described in the next section is being used to control the timestep.

3.2. Local Error Estimation. The aim of most integrators for o.d.e.s of the form of (10) is to control the local error per step so that it is less than a user-supplied tolerance, TOL i.e.

$$|| \ \underline{le}_{n+1}(t) \ || \ = \ \lambda \ TOL \ , \tag{20}$$

where λ is a positive acceptance factor that is less than one . In the case of the Theta Method the local error is given by

$$\underline{le}_{n+1}(t_{n+1}) = -\underline{y}_{n+1}(t_{n+1}) + \underline{V}(t_n) + (1-\theta)k\dot{\underline{V}}(t_n) + \theta k\underline{F}_M(t_{n+1}, \underline{V}(t_{n+1})).$$

Expanding $\underline{V}(t_n)$, ($= \underline{y}_{n+1}(t_n)$) about t_{n+1} using the local solution, as defined by (3.4), and its time derivatives gives:

$$\underline{le}_{n+1}(t_{n+1}) \;=\; \theta k\;(\underline{F}_M(t_{n+1},\underline{V}(t_{n+1}))\;-\;\underline{F}_M(t_{n+1},\underline{y}_{n+1}(t_{n+1}))\;)$$

$$-(1-\theta)k(\dot{\underline{y}}_{n+1}(t_{n+1})-\dot{\underline{V}}(t_n))+\frac{1}{2}k^2\underline{y}^{(2)}_{n+1}(t_{n+1})-\frac{k^3}{6}\underline{y}^{(3)}_{n+1})(t_{n+1})+O(k^4).$$

Applying the mean-value theorem to the terms involving $\underline{F}_M$, using the definitions (15) and expanding the derivatives of $\underline{y}_{n+1}(t_{n+1})$ about t_n gives

$$(I-k\theta J)\underline{le}_{n+1}(t_{n+1}) \;\approx\; (\theta-\frac{1}{2})k^2\underline{y}^{(2)}_{n+1}(t_n)+(\theta-\frac{1}{3})\frac{k^3}{2}\underline{y}^{(3)}_{n+1}(t_n)+O(k^4).$$

In the case when the o.d.e. system is stiff, the LU decomposition of the matrix ($I-k\;\theta\;J$) is available and may be used as in estimating the local error , see [3]. In the non-stiff case the term $k\theta J$ is assumed to be small, see (19), and so is discarded giving the local error estimate:

$$\underline{le}_{n+1}(t_{n+1})=(\theta-\frac{1}{2})k^2\underline{y}^{(2)}_{n+1}(t_n)+(\theta-\frac{1}{3})\frac{k^3}{2}\underline{y}^{(3)}_{n+1}(t_n)+O(k^4). \tag{21}$$

The implementation of this local error estimate used here is given by:

$$\underline{le}_{n+1}(t_{n+1})=(\theta-\frac{1}{2})\;k\;\underline{\Delta}_{n+1}\;\;+\;\;(\frac{s}{1+s}\;)\frac{1}{6}\;\;(\;k\;\underline{\Delta}_{n+1}\;-\;s\;k\;\;\underline{\Delta}_n\;) \tag{22}$$

where

$$s\;=\;\frac{t_{n+1}-t_n}{t_n-t_{n-1}}\;\;,\;\;\underline{\Delta}_n\;=\;(\dot{\underline{V}}(t_n)-\dot{\underline{V}}(t_{n-1}))\;,$$

$\underline{\Delta}_{n+1}$ is similarly defined and both vectors are stored by the code. A Taylor's series analysis gives

$$\underline{\Delta}_{n+1}\;=\;k\;\underline{V}^{(2)}(t_n)\;+\;\frac{k^2}{2}\;\;\underline{V}^{(3)}\;(t_n)\;+\;O(k^3).$$

and a similar expression for $\underline{\Delta}_n$. Substituting $\underline{\Delta}_{n+1}$ and $\underline{\Delta}_n$ into (22) gives an expression of the form of (21) but with the derivatives of the local solution replaced by those of the computed solution. An alternative error estimate, derived originally for stiff o.d.e.s, is used by Berzins and Furzeland [3].

The choice of error norm should reflect the accuracy being sought in the numerical solution as a whole. In the case of hyperbolic conservation laws this means that we must use an appropriate norm in which the method converges - such as an L_1 or an L_2 norm, Leveque [13]. In this paper the discrete L_1 norm in space will be used unless otherwise stated.

3.3. Stability for Advection Equation. This section will consider the issue of stability for the advection equation defined by

$$\frac{\partial u}{\partial t}\;+\;a\;\frac{\partial u}{\partial x}\;+\;b\;\frac{\partial u}{\partial y}\;=\;0\;,$$

with known initial and boundary values. A zero limiter (7) gives the standard first-order in space upwind discretization

$$\dot{U}_{j,k}(t) \;=\; -\frac{a}{\Delta x}\,[\,U_{j,k}(t) - U_{j-1,k}(t)\,] \quad -\frac{b}{\Delta y}\,[U_{j,k}(t) - U_{j,k-1}(t)]$$

which may be written in vector notation as the o.d.e. system:

$$\underline{\dot{U}}(t) \;=\; -\frac{a}{\Delta x}\,[\,A_N^x\,]\,\underline{U}(t) \;-\frac{b}{\Delta y}\,[\,A_N^y\,]\,\underline{U}(t)\;. \tag{23}$$

The spatial discretisation method of Section 2 on a uniform mesh gives rise to the o.d.e. system:

$$\begin{aligned}\dot{U}_{j,k}(t) = {} & \frac{-a}{\Delta x}[1+\frac{1}{2}(B_i(r_j^x,1) - \frac{1}{r_{j-1}^x}B_i(r_{j-1}^x,1))][U_{j,k}(t) - U_{j-1,k}(t)] \\ & \frac{-b}{\Delta y}\,[1+\frac{1}{2}(B_i(r_j^y,1) - \frac{1}{r_{j-1}^y}B(r_{j-1}^y,1))][U_{j,k}(t) - U_{j,k-1}(t)],\end{aligned} \tag{24}$$

where $i = 1, 2$ or 3 denotes the limiter in use. Using vector notation this may be written as:

$$\underline{\dot{U}}(t) = -\frac{a}{\Delta x}[I + B_i^x(t,\underline{U}(t))\,]\;A_N^x\;\underline{U}(t) - \frac{b}{\Delta y}[I + B_i^y(t,\underline{U}(t))\,]\;A_N^y\;\underline{U}(t) \tag{25}$$

where the matrix A_N^x corresponds to the standard first order upwind approximation in the x dimension and B_i^x corresponds to the terms involving the limiter function $B_i(.,.)$ in the x dimension. For the sake of brevity the dependence of the matrices B_i^x and $\mathrm{B_i^y}$ on the solution will be assumed but no longer stated. Define the matrix C_N^x to be the product of these matrices:

$$C_N^x \;= [\;I \;+\; B_i^x\;]\;A_N^x\;,$$

and define the matrix C_N^y analogously so that equation (25) may be written as.

$$\underline{\dot{U}} \;=\; -\frac{a}{\Delta x}\;C_N^x\;\;\underline{U}(t) \;-\frac{b}{\Delta y}\;C_N^y\;\;\underline{U}(t) \tag{26}$$

Assuming that the dependence of the matrices C_N^x and C_N^y on the solution through the limiter terms may be neglected, the stability limit for functional iteration to converge for this problem with the time integration and spatial discretisation methods used is, see equation (19),

$$k\;\theta\;||\;\frac{a}{\Delta x}\;C_N^x\;+\frac{b}{\Delta y}\;C_N^y\;||\;\;<\;\;r_c\;\;. \tag{27}$$

Define the ratio δ by $\delta = \frac{|a|\Delta y}{|b|\Delta x}$. The addition of equation (27) to itself multiplied by δ then gives

$$k\;\left[\frac{|a|}{\Delta x}\;+\;\frac{|b|}{\Delta y}\right]||\;C_N^x\;+\;\delta\;C_N^y\;||\;<\;\frac{r_c(1+|\delta|}{\theta}\quad. \tag{28}$$

The requirement that functional iteration converges thus imposes a CFL type stability condition. While this CFL condition provides a stable timestep there is no guarantee of accuracy. It will be shown that the new accuracy control introduced here also gives rise to a similar condition.

3.4. Integration using Stability Criteria Only. A standard approach in the time integration of hyperbolic equations is to use the CFL stability condition alone to choose the timestep. This approach may also be used with the Theta Method described above. In this case (assuming $\Delta x = \Delta y$) the time step is chosen solely so that it satisfies the equation

$$\frac{k}{\Delta x} \ < \ CFL \ . \tag{29}$$

with the number CFL specified by the user. In the experiments reported in Section 6 two functional iterations are performed on each time step. Great care has to be taken when chosing the value of CFL as the numerical solution may become unstable for values that are too large.

4. Temporal Error Control and Balancing Space and Time Errors. The main difficulty with the temporal local error per step (LEPS) control is that the time global error is not proportional to the local error tolerance, *tol* . In order to balance the spatial and temporal errors, it is desirable that the error control strategy should yield a solution with a time global error that is proportional to the requested accuracy. This is said to be the case, Higham [8], if the global error at time t for an accuracy requirement TOL is $g(t)_{TOL}$, then, for $r \ > \ 0$,

$$g(t)_{TOL \times r} \ = \ r \ \times \ g(t)_{TOL} \ .$$

Higham [8] describes the work of Stetter who shows that this tolerance proportionality is satisfied for all t_n if and only if the local errors $\underline{le}_{n+1}(t)$ obtained for an accuracy tolerance *tol* satisfy

$$\underline{le}_{n+1}(t_{n+1}) = \underline{\bar{\gamma}} \ (t_{n+1}, t_n) \ k_n \ tol + o(\ k_n tol \ ), \ n = 0, ..., p-1 \tag{30}$$

where $\underline{\bar{\gamma}}(\tau, t)$ behaves like an integral mean over $[\tau, t]$ of a function that is independent of *tol* and bounded on $[0, t_e]$ and $k_n \ = t_{n+1} \ - \ t_n$. Here, $o(k_n tol)$ denotes a term that is numerically negligible compared with terms of order $k_n tol$ in the same equation. From this it follows that in order that the time global error be a fraction of the spatial discretisation error, i.e. proportional to it, we must control the LEPUS rather than the LEPS, with respect to a tolerance based on the spatial discretisation error. This may be achieved by considering the relative contributions of the temporal local error and the spatial discretisation error to the overall error in the numerical solution. The equation for the evolution of the spatial discretisation error is, [11], after neglecting second-order terms in the error,

$$\dot{\underline{e}}(t) \ = \ J \ \underline{e}(t) + \underline{TE}(t, \underline{u}(t) \ ), \quad \underline{e}(0) = \underline{0}, \tag{31}$$

where the space truncation error is defined by

$$\underline{TE}(t, \underline{u}(t) \ ) \ = \ \dot{\underline{u}} - \underline{F}_M(t, \underline{u}(t)) \tag{32}$$

and where the Jacobian matrix, J , is defined by $J \ = \ \left[\frac{\partial \underline{F}_M}{\partial \underline{V})} \right]$. Integrating equation (31) over the time step $[t_n, t_{n+1}]$ gives the growth of the spatial discretisation error

$$\underline{e}(t_{n+1}) - \underline{e}(t_n) = \int_{t_n}^{t_{n+1}} J \ \underline{e}(t) \ + \ \underline{TE}(t) \ dt \ .$$

The growth of the global time error over the same interval is governed by a similar equation (see [11])

$$\underline{g}(t_{n+1}) - \underline{g}(t_n) \ = \underline{le}_{n+1}(t_{n+1}) + \int_{t_n}^{t_{n+1}} J\underline{W}(t)\ dt,$$

where $\underline{W}(t_n) = \underline{g}(t_n)$. Thus, the growth of the overall error in the method of lines is governed by

$$\underline{E}(t_{n+1}) - \underline{E}(t_n) = \underline{le}_{n+1}(t_{n+1}) + \int_{t_n}^{t_{n+1}} J(\underline{v}(t) + \underline{W}(t)) \ + \ \underline{TE}(t)\ dt \tag{33}$$

where $\underline{E}(t_n) = \underline{v}(t_n) + \underline{W}(t_n)$ is the global error at time t_n. This is the key equation for balancing the spatial and temporal errors, as it shows how the time local error and the spatial truncation error both contribute to the growth of the overall error. Lawson et al [11] show that the time error will not be dominant if an error control of the form:

$$|| \ \underline{le}_{n+1}(t_{n+1}) \ || \ \leq \ \epsilon \ || \ \int_{t_n}^{t_{n+1}} J(\underline{v}(t) + \underline{W}(t)) + \underline{TE}(t) \ \ dt \ \ || \tag{34}$$

is used. Furthermore the presence of the integral on the right-hand side of this equation gives a natural LEPUS approach. Lawson et al [11] implement this strategy by estimating the overall error using a global error estimator which depends for its efficiency on the main integration using a modified Newton method. Their approach is thus not so attractive for convection-dominated problems for which explicit time integration methods or implicit methods using functional iteration are more likely to be used.

4.1. Balancing the Local Space and Time Contributions. The simplest alternative approach is to balance the local space and time contributions to the error. This approach probably originates with the work of Gary [6], who used it to determine the optimal time step for fixed time step methods. Neglecting two out of the three terms in the integral on the right side of equation (34) gives rise to the following error control in the o.d.e. time integration:

$$|| \ \hat{\underline{le}}_{n+1}(t_{n+1}, tol) \ || \ < \quad \epsilon \ k_n \ || \ \underline{TE}(t_{n+1}) \ || \tag{35}$$

where the vector norm used is that of Section 3 and the integral of the space truncation error is approximated by a one-point quadrature rule (and the LEPUS accuracy tolerance *tol* is given by $\epsilon \ || \ \underline{TE}(t_{n+1}) \ ||$, in this case).

4.2. An Error Control Strategy based on Moore and Flaherty. Moore and Flaherty [14] use an error control algorithm for parabolic equations with the advantages of both the above methods in that the estimate is local to the timestep but also provides a measure of the error in the computed solution. The algorithm is local in that it assumes that at the start of any timestep the error is zero. The disadvantage of this approach is that there is no overall estimate of the combined spatial and temporal error, such as that of Strouboulis and Oden [15]. Equation (33) provides a way of comparing this approach with that of Lawson et. al. [11]. The quantity that is estimated, $\underline{\hat{e}}(t)$, may be referred to as the local in time space error and may be written as:

$$\underline{\hat{e}}(t_{n+1}) = \int_{t_n}^{t_{n+1}} J \ \underline{\hat{e}}(t) \quad + \underline{TE}(t)\ dt \tag{36}$$

where $\hat{\underline{e}}(t_n) = 0$. The Moore and Flaherty error control is then of the form given by

$$|| \underline{le}_{n+1}(tol) || \; < \; \epsilon \; || \hat{\underline{e}}(t_{n+1}) ||. \tag{37}$$

By using equation (36) equation (33) may be written as

$$\underline{E}(t_{n+1}) - \underline{E}(t_n) - \underline{le}_{n+1}(t_{n+1}) \; = \; \hat{\underline{e}}(t_{n+1}) \; + \; \int_{t_n}^{t_{n+1}} J \; \underline{W}(t) \; \; dt$$

where $\underline{W}(t_n) \; = \; \underline{E}(t_n)$. From this we see that the Moore and Flaherty error control has similarities with that of Lawson et. al. [11]. The main difference between the two is that the Moore and Flaherty approach neglects part of the integral term in equation (33). This term corresponds to the propagation forward over the time step to t_{n+1} of the existing error at time t_n . Providing that the integration is stable this term can reasonably be expected to decrease in size.

The implementation of this estimate used here differs from that of [14] in that the different choices of limiters defined by equations (7) to (9) allow three different options regarding computing the solution and estimating the error.
Method A: In this case the computed solution is the second-order solution (8) and the first-order solution (7) is computed for use in error estimation.
Method B: The computed solution is again the second-order solution (8) and a third-order solution (9) is calculated for error estimation.
Method C: The computed solution is the first-order solution (7) and a second-order solution (8) is calculated for error estimation.

All three methods involve the calculation of a main solution and an auxiliary calculation to generate a solution of different order for use in spatial error estimation. In the case of Method A the difference between the two solutions is only an estimate of the error in the low-order solution and so local extrapolation in space is effectively being used when we move forward in time with the high-order solution. The alternative (Method C) is to compute a first-order solution and then to estimate the error by computing a second-order solution. It will be seen that this reduces the accuracy by a factor of two, see Section 6, compared to Method A as it is the first-order error that is carried forward rather than the second-order error. It is perhaps worth stressing here that for problems consisting only of shock-wave type solutions, both the $first$ and *second* order methods give solutions which are first-order accurate.

An alternative possibility is to use an h-extrapolation approach to estimate the error e.g. Berger and Oliger [1] and Lawson et al [11].

Regardless of which method is used, the auxiliary solution used in spatial error estimation is assumed to be computed by a modified o.d.e. system which will be denoted by

$$\dot{\underline{v}}_{n+1}(t) = \hat{\underline{F}}_M(t, \underline{v}_{n+1}(t)), \;\; \underline{v}_{n+1}(t_n) = \underline{V}(t_n), \;\; \dot{\underline{v}}_{n+1}(t_n) = \hat{\underline{F}}_M(t_n, \underline{V}(t_n)). \tag{38}$$

The local-in-time space error is then estimated from

$$\hat{\underline{e}}(t_{n+1}) \; = \; \underline{V}(t_{n+1}) \; - \; \underline{v}_{n+1}(t_{n+1}). \tag{39}$$

The truncation error of the auxiliary solution is defined by

$$\underline{TE}_l(t, \underline{u}(t) \;) \; = \; \dot{\underline{u}}(t) \; - \hat{\underline{F}}_M(t, \underline{u}(t)) \tag{40}$$

and the spatial error of the auxiliary solution, denoted by $\underline{e}_l(t)$, satisfies an equation similar to equation (31), after again neglecting second order terms:

$$\dot{\underline{e}}_l(t) \;=\; J_l\ \underline{e}_l(t) + \underline{TE}_l(t, \underline{u}(t)\ )\ \ ,\ \ \underline{e}_l(0) = \underline{0}, \tag{41}$$

where the Jacobian matrix J_l is defined by

$$J_l \;=\; \frac{\partial \underline{\hat{F}}_M}{\partial \underline{V}}\ .$$

4.3. Tolerance Proportionality. The aim of this section is to give a brief and heuristic argument to indicate that the spatial error dominates the temporal error. The main idea is that if the spatial error is proportional to some mesh parameter, say Δx , then the temporal error will, when using the error control (37), be proportional to a tolerance $\epsilon\ \Delta x$. Hence for a suitably chosen ϵ the spatial error will dominate. In order to demonstrate this tolerance proportionality the first step is to analyze the error control assuming that there is no temporal error. The imposition of the condition

$$\underline{e}_l(t_n) \;=\; \underline{e}(t_n)$$

allows the local in time space error defined by (39), assuming that there is no temporal error, to be written as

$$\underline{\hat{e}}(t_{n+1}) \;=\; \underline{e}_l(t_{n+1}) \;-\; \underline{e}(t_{n+1}) \tag{42}$$

Combining equations (31), (41) and (42) gives an equation for the the growth of this error over the time step $[t_n, t_{n+1}]$:

$$\underline{\hat{e}}(t_{n+1}) = \int_{t_n}^{t_{n+1}} J_l \underline{\hat{e}}(s) - [J - J_l]\underline{e}(s) - \underline{TE}(s, \underline{u}(s)\ ) + \underline{TE}_l(s, \underline{u}(s)\ )ds, \tag{43}$$

where $\underline{\hat{e}}(t_n) = 0$. From the theory of Coppel, as used by Lawson et al [11], this may be rewritten as

$$\underline{\hat{e}}(t_{n+1}) = Y^*(t_{n+1}) \int_{t_n}^{t_{n+1}} [Y^*]^{-1}(s)\ [[J_l - J]\underline{e}(s) - \underline{TE}(s, \underline{u}(s)\ )) + \underline{TE}_l(s, \underline{u}(s)\ )]\ ds, \tag{44}$$

where $Y^*(t)$ is the fundamental matrix, [11], for the equation $\dot{\underline{v}} = J_l\ \underline{v}$. The spatial error in the primary solution, $\underline{e}(s)$, appearing in this equation may similarly be written as

$$\underline{e}(t) \;=\; Y(t)Y^{-1}(t_n)\underline{e}(t_n) + Y(t) \int_{t_n}^{t}\ Y^{-1}(s)\underline{TE}(s, \underline{u}(s)\ )ds \tag{45}$$

where $Y(t)$ is the fundamental matrix for the equation $\dot{\underline{v}} = J\underline{v}$, [11]. Equations (44) and (45) show that the error $\underline{\hat{e}}(t_{n+1})$ can be written in terms of the two spatial truncation errors and the spatial error at t_n. The assumption in a tolerance proportionality argument for spatial error domination is that it is possible to extract a constant factor from both truncation errors. For instance in the case of a first order method the truncation error may be assumed to have the form

$$\underline{TE}(t) \;=\; \Delta x\ \underline{TE}_1^\sharp(t) \;+\; \Delta y\ \underline{TE}_2^\sharp(t)\ \ .$$

By combining the two right-hand terms and extracting a common factor of Δx , assuming that the ratio $\Delta x/\Delta y$ is a constant it is possible to rewrite the above equation in the form:

$$\underline{TE}(t) \;=\; \Delta x \; \underline{TE}^{\sharp}(t) \;\; . \tag{46}$$

The same approach may be adopted for the auxiliary truncation error

$$\underline{TE}_l(t) \;=\; \Delta x \; \underline{TE}_l^{\sharp}(t) \; .$$

This assumption will be shown to hold for model problems in Section 5 below. From equation (45) it also follows that this common factor may also be extracted from the spatial error

$$\underline{e}(t) \;=\; \Delta x \; \underline{e}^{\sharp}(t) \; .$$

Given this assumption the temporal error control may be written as

$$||\underline{le}_{n+1}(t_{n+1})|| \;\leq \qquad \epsilon \; \Delta x \; \times$$

$$||Y^*(t_{n+1}) \int_{t_n}^{t_{n+1}} [Y^*]^{-1}(s) \left[[J - J_l]\underline{e}^{\sharp}(s) + \underline{TE}^{\sharp}(s, \underline{u}(s)\;)) - \underline{TE}_l^{\sharp}(s, \underline{u}(s)\;) \right] ds||.$$

Thus, the error control strategy is of the required LEPUS form, [8], given by equation (30):

$$||\; \underline{le}_{n+1}(t_{n+1}) \;|| \;\leq\; TOL \;\times\; k \;\times\; \frac{1}{t_{n+1} - t_n} \; \int_{t_n}^{t_{n+1}} \; \underline{\gamma}(t) \; dt$$

where $\underline{\gamma}(t)$ does not depend on TOL , which here is $\epsilon \;\Delta x$, and $k \;=\; t_{n+1} - t_n$. This may be used as in Lawson et al [11] as part of a proof by induction to show, using equation (46) and Stetter's results, that the global error due to time integration will be proportional to $\epsilon \; \Delta x$ while from (46) the spatial error will be proportional to Δx. Thus for ϵ sufficiently small the spatial error dominates.

5. Implementation and Analysis of Error Balancing. This section describes how error balancing is implemented in a code and provides an analysis of this implementation.

5.1. Computing the Time Tolerance. The local in time space error is computed by applying the θ method of Section 3 to equation (38) to get

$$\underline{v}_{n+1}(t_{n+1}) = \underline{V}(t_n) + \theta k \hat{\underline{F}}_M(t_{n+1}, \underline{v}_{n+1}(t_{n+1})) + (1-\theta) k \hat{\underline{F}}_M(t_n, \underline{V}(t_n)).$$

In practice, as only the order of magnitude of the norm of the spatial error is needed, it appears sufficient to use only one step of functional iteration to compute $\underline{v}_{n+1}(t_{n+1})$ with $\underline{V}(t_{n+1})$ being used as the predicted value. Combining this with equations (17) and (39) gives

$$\begin{aligned} \hat{\underline{e}}(t_{n+1}) \;=\; \theta \quad k \quad & [\; \underline{F}_M(t_{n+1}, \underline{V}(t_{n+1})) \;-\; \hat{\underline{F}}_M(t_{n+1}, \underline{V}(t_{n+1})) \;] \\ +(1-\theta) \quad k \quad & [\; \underline{F}_M(t_n, \underline{V}(t_n)) \;-\; \hat{\underline{F}}_M(t_n, \underline{V}(t_n)) \;] \end{aligned} \tag{47}$$

where $k = t_{n+1} - t_n$. Assuming that the main solution is calculated by functional iteration to convergence then

$$\underline{\dot{V}}(t_n) = \underline{F}_M(t_n, \underline{V}(t_n)) \quad . \tag{48}$$

In the case when the primary solution and its time derivative are substituted into equation (38), denote the residual of the secondary integration by

$$\underline{r}\ (t_n, \underline{V}(t_n)) = \underline{\dot{V}}(t_n) - \underline{\hat{F}}_M(t_n, \underline{V}(t_n)) \ . \tag{49}$$

Combining equations (48) and (49) enables equation (47) to be rewritten as

$$\underline{\hat{e}}(t_{n+1}) = \theta k\ \ \underline{r}(t_{n+1}, \underline{V}(t_{n+1})) + (1-\theta)k\ \ \underline{r}(t_n, \underline{V}(t_n)) \ . \tag{50}$$

From this equation the LEPUS tolerance used in the code on the step to $t_{n+1}$is then given by

$$TOL = \epsilon\ \ || \ \theta\ \underline{r}(t_{n+1}, \underline{V}(t_{n+1})) + (1-\theta)\ \underline{r}(t_n, \underline{V}(t_n)) \ \ || \ .$$

An LEPS tolerance may also be defined from (50) by

$$TOL = \epsilon\ \ || \ \underline{\hat{e}}(t_{n+1}) \ || \tag{51}$$

and may be used equally well in a LEPS code.

5.2. The Effect of Errors on TOL. It is necessary to consider the effect of the errors in the solution on the integrity of the error estimator defined by (47). From equation (14) it follows that

$$\underline{V}(t_{n+1}) = \underline{u}(t_{n+1}) - \underline{E}(t_{n+1}) \ .$$

This expression is substituted into equation (47) . The mean value theorem is applied and the truncation errors of the primary and secondary discretisations as defined by equations (40) and (32) respectively, are used to rewrite equation (47) as

$$\begin{aligned} \underline{\hat{e}}(t_{n+1}) = \ & - \ \theta k[\ \underline{TE}(t_{n+1}, \underline{u}(t_{n+1})) - \underline{TE}_l(t_{n+1}, \underline{u}(t_{n+1})) + \underline{G}(t_{n+1})\ ] \\ & - \ (1-\theta)k[\ \underline{TE}(t_n, \underline{u}(t_n)) - \underline{TE}_l(t_n, \underline{u}(t_n)) \ + \ \underline{G}(t_n)\ ] \end{aligned} \tag{52}$$

where

$$\underline{G}(t) = [J - J_l]\ \underline{E}(t). \tag{53}$$

This equation may be seen to be related to the integral on the right side of (44) and also shows that the effect of any time global error in the determination of TOL is multiplied by $\theta\ \epsilon$. From the previous section it is also to be expected that the norm of the time global error is a fraction ϵ of the norm of the spatial error. For ϵ sufficiently small it follows that the value of TOL should not be corrupted by temporal error. This result is similar to that proved by Lawson et al [11] for their error estimator. Equations (37) and (52) show that the new error control, when implemented using (47), reduces to one very similar to that of Gary [6] in the fixed step case.

The error estimator for $\underline{\hat{e}}(t)$ is very similar in the case when the low-order solution is the accepted solution and the high-order solution is only computed locally for use

in the error estimate. A straightforward analysis along the lines of that above leads to

$$\begin{aligned}(54)\qquad \underline{\hat{e}}(t_{n+1}) = &- \theta k[\ \underline{TE}(t_{n+1},\underline{u}(t_{n+1})) - \underline{TE}_l(t,\underline{u}(t_{n+1})) + \hat{\underline{G}}(t_{n+1})\ ] \\ &- (1-\theta)k[\ \underline{TE}(t_n,\underline{u}(t_n)) - \underline{TE}_l(t_n,\underline{u}(t_n)) + \hat{\underline{G}}(t_n)\ ]\end{aligned}$$

where

$$(55)\qquad \hat{\underline{G}}(t) \ = \ [J \ - \ J_l] \ \underline{E}_l(t)\ .$$

and where $\underline{E}_l(t_{n+1})$ is now the global error in the low order solution. A comparison of equations (52) and (53) with equations (54) and (55) shows that the propagated error on the right side of (54) is the high-order error while in (52) it is the low order error. The numerical results in Section 6 will show that this difference can make a large change to the dynamic behaviour of the error.

5.3. Order of Truncation Error. This section will examine the form of the error estimator for the advection equation of Section 3.3. From equations (23) (24) and (47) the local-in-time spatial error for methods B and C as defined in Section 4.2 has the form

$$\begin{aligned}(56)\underline{\hat{e}}(t_{n+1}) = \quad & k \quad \theta(\frac{-a}{\Delta x}[B_i^x - B_{i+1}^x]A_N^x\underline{V}(t_{n+1}) - \frac{b}{\Delta y}[B_i^y - B_{i+1}^y]A_N^y\underline{V}(t_{n+1})) \\ + \quad & k \quad (1-\theta)(\frac{-a}{\Delta x}[B_i^x - B_{i+1}^x]A_N^x\underline{V}(t_n) - \frac{b}{\Delta y}[B_i^y - B_{i+1}^y]A_N^y\underline{V}(t_n))\ .\end{aligned}$$

It should be noted that the right side of this equation is multiplied by (-1) if Method A of Section 4.2 is used. All the four terms on the right side of this expression are of the same type. Consider for example the term

$$(57)\qquad \frac{a}{\Delta x}\ [\ B_i^x \ - \ B_{i+1}^x\ ]\ A_N^x\ \underline{V}(t_{n+1})\ .$$

Denote the components of the vector $\underline{V}_{n+1}$) as in equation (11). Define

$$\Phi_j \ = \ V_{j+1,k}(t) - V_{j,k}(t)\ ,\ \ \Phi_{j-1} \ = \ V_{j,k}(t) - V_{j-1,k}(t)\ ,\ \ \Phi_{j-2} \ = \ V_{j-1,k}(t) - V_{j-2,k}(t)\ ,$$

$$(58)\qquad r_j^x \ = \ \Phi_j/\Phi_{j-1} \text{ and } \mathrm{r_{j-1}^x} \ = \ \Phi_{\mathrm{j-1}}/\Phi_{\mathrm{j-2}}\ \ .$$

From standard Taylor's series analysis it follows that

$$\Phi_j - \Phi_{j-1} \ = \ O(\Delta x^2) \quad \text{and } \Phi_{\mathrm{j-1}} - \Phi_{\mathrm{j-2}} \ = \ \mathrm{O}(\Delta \mathrm{x}^2)\ .$$

Using this notation each component of equation (57) has the form

$$\frac{a}{2\Delta x}[B_i(r_j^x,1) - B_{i+1}(r_j^x,1) - \frac{B_i(r_{j-1}^x,1)}{r_{j-1}^x} + \frac{B_{i+1}(r_{j-1}^x,1)}{r_{j-1}^x}][V_{j,k}(t) - V_{j-1,k}(t)].$$

5.3.1. Method A Error Estimator. In the case of Method A, as defined in Section 4.2 , $i = 1$ in equation (57) . Substituting in from (7), and (8) enables the term in (57) to be written as

$$-\ \frac{a}{2\Delta x}\ [\frac{\Phi_j|\Phi_{j-1}| + \Phi_{j-1}|\Phi_j|}{|\Phi_j| + |\Phi_{j-1}|}\ -\ \frac{\Phi_{j-1}|\Phi_{j-2}| + \Phi_{j-2}|\Phi_{j-1}|}{|\Phi_{j-1}| + |\Phi_{j-2}|}\ ]\ .$$

This may be rewritten as

$$(59)\quad -\frac{a}{2\Delta x}\,[\,\frac{|\Phi_{j-1}|}{|\Phi_j|+|\Phi_{j-1}|}\,(\Phi_j-\Phi_{j-1})\;+\;\frac{|\Phi_{j-1}|}{|\Phi_{j-1}|+|\Phi_{j-2}|}\,(\Phi_{j-1}-\Phi_{j-2})]\,.$$

Define the terms $0<\alpha_j<1$ and $0<\beta_j<1$ by

$$\alpha_j\;=\;\frac{|\Phi_{j-1}|}{|\Phi_j|+|\Phi_{j-1}|}\quad,\quad\beta_j\;=\;\frac{|\Phi_{j-1}|}{|\Phi_{j-2}|+|\Phi_{j-1}|}\,.$$

Then equation (59) may be written as

$$-\,\frac{a}{2\Delta x}\,[\alpha_j\,(\Phi_j-\Phi_{j-1})\;+\;\beta_j\,(\Phi_{j-1}-\Phi_{j-2})]\,.$$

From definitions (58) this term is $O(\Delta x)$ and so by applying the same approach as above to all the terms in equation (56) the error estimator $\hat{\underline{e}}(t_{n+1})$, as defined by equation (50), may be rewritten as

$$(60)\qquad \hat{\underline{e}}(t_{n+1})\;=\;k\,[\,\underline{E}_1(t_n,t_{n+1})\,\Delta x\;+\;\underline{E}_2(t_n,t_{n+1})\,\Delta y\,]$$

where the vectors $\underline{E}_1(t_n,t_{n+1})$ and $\underline{E}_2(t_n,t_{n+1})$ collect together the terms at time t_n and t_{n+1} .

5.3.2. Method B Error Estimator. In the case of Method B as defined in Section 4.2 $i=2$ in (57) and substituting in from (8), and (9) into equation (57) gives

$$\frac{a}{2\Delta x}\,[\,\frac{\Phi_j|\Phi_{j-1}|+\Phi_{j-1}|\Phi_j|}{|\Phi_j|+|\Phi_{j-1}|}\;-\;\frac{(3\Phi_j+\Phi_{j-1})}{4}\;-\;\frac{\Phi_{j-1}|\Phi_{j-2}|+\Phi_{j-2}|\Phi_{j-1}|}{|\Phi_{j-1}|+|\Phi_{j-2}|}\;+\;\frac{(3\,\Phi_{j-1}+\Phi_{j-2})}{4}\,]\,.$$

This may be rewritten as

$$\frac{a}{2\Delta x}\,[\,(\frac{|\Phi_{j-1}|}{|\Phi_j|+|\Phi_{j-1}|}\;-\;\frac{3}{4}\,)\,(\Phi_j-\Phi_{j-1})\;+\;(\;\frac{|\Phi_{j-1}|}{|\Phi_{j-1}|+|\Phi_{j-2}|}\;-\;\frac{1}{4}\,)(\Phi_{j-1}-\Phi_{j-2})\,]$$

and again as

$$\frac{a}{4\Delta x}[\frac{|\Phi_{j-1}|-|\Phi_j|}{|\Phi_j|+|\Phi_{j-1}|}(\Phi_j-\Phi_{j-1})+\frac{|\Phi_{j-1}|-|\Phi_{j-2}|}{|\Phi_{j-1}|+|\Phi_{j-2}|}(\Phi_{j-1}-\Phi_{j-2})-\frac{1}{2}(\Phi_j-2\Phi_{j-1}+\Phi_{j-2})].$$

There are two cases to consider. In the case when Φ_j and Φ_{j-1} and Φ_{j-2} have the same sign (say negative- purely for illustration) then the van Leer method is second order and the term may be rewritten as

$$\frac{a}{4\Delta x}\,\left[\;\frac{(\Phi_j-\Phi_{j-1})^2}{|\Phi_j|+|\Phi_{j-1}|}\;-\;\frac{(\Phi_{j-1}-\Phi_{j-2})^2}{|\Phi_{j-1}|+|\Phi_{j-2}|}\;-\;\frac{1}{2}(\Phi_j-2\,\Phi_{j-1}+\Phi_{j-2})\;\right]\,.$$

Straightforward Taylor's series analysis shows that all the terms within [...] are $O(\Delta x^3)$ and hence that the error estimate will have the form

$$\hat{\underline{es}}(t_{n+1})\;=\;k\,[\,\underline{E}_3(t_n,t_{n+1})\,\Delta x^2\;+\;\underline{E}_4(t_n,t_{n+1})\,\Delta y^2\,]\,.$$

where the vectors $\underline{E}_3(t_n, t_{n+1})$ and $\underline{E}_4(t_n, t_{n+1})$ collect together the terms at time t_n and t_{n+1} . In the case when (say) the signs of Φ_j and Φ_{j-1} differ then there is a cancellation resulting from the equality

$$\frac{(\Phi_j - \Phi_{j-1})}{|\Phi_j| + |\Phi_{j-1}|} \;=\; +1 \;\text{ or }\; -1 \;,$$

and hence the error estimate will have the same form as (60). Numerical testing on the test problems in Section 6 shows that there is less difference than might be expected between using Method A and Method B. In both cases the van Leer limiter reverts to first-order for two thirds of the cases when non-zero solution gradients are present. The overall rate of convergence in the L_1 norm is only first order in both cases.

5.4. Stability and Accuracy Requirements. As the error control (37) takes into account the local growth in both the spatial and temporal errors it is of interest to understand the relationship between this error control and the stability requirement (19). Substituting equation (22) for the time local error and equation (47) for the local in time space error into the error balancing equation (37) gives, assuming constant timestep, the following equation:

$$|| \; (\theta - \frac{1}{2}) \;\; k \; (\dot{\underline{V}}(t_{n+1}) - \dot{\underline{V}}(t_n)) \;+\; \frac{k}{12} \; (\dot{\underline{V}}(t_{n+1}) - 2 \; \dot{\underline{V}}(t_n) + \; \dot{\underline{V}}(t_{n-1})) \; ||$$

$$= \; \lambda \;\; \epsilon \;\; k \; ||\theta \; \underline{r}(t_{n+1}, \; \underline{V}(t_{n+1})) + (1 - \theta) \; \underline{r}(t_n, \; \underline{V}(t_n) \;)|| \tag{61}$$

where $0 < \lambda < 1$ is the value in the local error acceptance test.

Assuming that the o.d.e. system may be treated as linear with a constant Jacobian, J , over the last two timesteps, then from the definition of $\underline{\Delta}_n$ in equation (22) it follows that

$$\underline{\Delta}_n \;=\; J \; (\underline{V}(t_n) - \underline{V}(t_{n-1})) \; .$$

From equation (17), this in turn may be written as

$$\underline{\Delta}_n \;=\; k \; J \; \underline{V}^*_n \;\text{ where }\; \underline{\mathrm{V}}^*_{\mathrm{n}} = (1 - \theta) \;\; \dot{\underline{\mathrm{V}}}(\mathrm{t_{n-1}}) \;\; + \theta \; \dot{\underline{\mathrm{V}}}(\mathrm{t_n}).$$

The left side of equation (61) may be rewritten as

$$|| \; (\theta - \frac{1}{2}) \; k \; (\dot{\underline{V}}(t_{n+1}) - \dot{\underline{V}}(t_n)) \;+\; \frac{1}{12} \; (\; k \; (\dot{\underline{V}}(t_{n+1}) - 2 \; \dot{\underline{V}}(t_n) + \; \dot{\underline{V}}(t_{n-1})) \; ||$$

$$= \; || \; k^2 \; J \; ((\theta - \frac{1}{2}) \; \underline{V}^*_{n+1} \;+\; \frac{1}{12} \; (\underline{V}^*_{n+1} \;-\; \underline{V}^*_n) \;) \; ||.$$

From a property of matrix norms:

$$\leq \; || \; J \; || \; k^2 \;\; || \; (\theta - \frac{1}{2}) \;\; \underline{V}^*_{n+1} \;+\; \frac{1}{12} \; (\underline{V}^*_{n+1} \;-\; \underline{V}^*_n) \;) \; || \;\; . \tag{62}$$

From equation (19) the convergence requirement on functional iteration means that

$$k \; || \; J \; || \;\leq\; r_c \;/\; \theta \; .$$

From equations (61) and (62) it follows that

$$k\ \lambda\ \epsilon\ \ ||\ \theta\ \underline{r}(t_{n+1},\underline{V}(t_{n+1}))+(1-\theta)\ \underline{r}(t_n,\underline{V}(t_n))\ ||$$

$$\leq\ r_c\ \frac{k}{\theta}\ ||(\theta-\frac{1}{2})\ \underline{V}^*_{n+1}\ +\ \frac{1}{12}\ (\underline{V}^*_{n+1}-\underline{V}^*_n))||$$

and so the product $\lambda\ \epsilon$ must satisfy

$$\lambda\ \epsilon\ \leq\ \frac{r_c\ ||\ (\theta-\frac{1}{2})\ \ \underline{V}^*_{n+1}\ +\ \frac{1}{12}\ (\underline{V}^*_{n+1}\ -\ \underline{V}^*_n)\ )\ ||}{\theta\ ||\ \theta\ \underline{r}(t_{n+1},\underline{V}(t_{n+1}))+(1-\theta)\ \underline{r}(t_n,\underline{V}(t_n))\ ||}\ . \tag{63}$$

This result shows that if ϵ is large (or the righthand side is small) then λ must be small enough to satisfy (63), the step size will be such that the error test is readily passed and stability will be controlling the time step size. This situation is readily detected in time integration when the code consistently computes small (< 0.1) values of λ as defined by (20) but is unable to increase the stepsize because of the restriction imposed by (19) . In this case decreasing ϵ will allow λ to grow.

It is natural to ask if the new error control imposes a Courant-type condition on the timestep. Equation (61) may be written as

$$k\ \frac{||\ (\theta-\frac{1}{2})\ J\ \underline{V}^*_{n+1}\ +\ \frac{1}{12}\ J\ (\underline{V}^*_{n+1}\ -\ \underline{V}^*_n)\ )\ \ ||}{||\theta\ \underline{r}(t_{n+1},Vu(t_{n+1}))+(1-\theta)\ \underline{r}(t_n,\underline{V}(t_n))\ ||}\ =\ \lambda\ \ \epsilon$$

Using the form of the error estimator calculated in the previous sub-section this may be written as

$$\frac{k}{||\underline{E}_1(t_n,t_{n+1})\Delta x+\underline{E}_2(t_n,t_{n+1})\Delta y||}=\frac{\lambda\ \ \epsilon}{||(\theta-\frac{1}{2})J\underline{V}^*_{n+1}+\frac{1}{12}J(\underline{V}^*_{n+1}-\underline{V}^*_n))||}$$

where $\underline{E}_1$ and $\underline{E}_2$ are known vectors whose precise form depends on whether method A or B is used. This equation shows that the new error control has a form of CFL stability condition built in and that the allowed timestep is proportional to ϵ .

6. Numerical Experiments. A number of experiments with problems in one space dimension using the error control strategy given by equation (37) above are given by Berzins [2] who also explains how the Theta Method time integrator was modified to incorporate the new control. The test problems include the standard Euler Equations Shock-Tube test problem of Sod, often used in the comparison of algorithms for hyperbolic equations. The intention here is to use four test problems to compare new approaches defined by Methods A, B and C in Section 4 with a standard local error control and with stability alone controlled integration for two dimensional problems. The four test problems used are:
1. Burgers' Problem I. A convection diffusion problem, known as Burgers' Equation, which is defined by

$$\frac{\partial u}{\partial t}\ +\ w(x,t)\ \frac{\partial u}{\partial x}\ +\ w(y,t)\ \frac{\partial u}{\partial y}\ -\ \nu\ (\frac{\partial^2 u}{\partial x^2}\ +\ \frac{\partial^2 u}{\partial y^2})\ =\ 0\ ,\ \nu=0.0001\ ,$$

$$\text{where}\quad (\mathrm{x},\mathrm{y},\mathrm{t})\ \epsilon\ (0,1)\ \times\ (0,1)\ \times\ (0,1]\ .$$

The analytic solution is given by $u(x,y,t) = w(x,t)\,w(y,t)$, where $w(x,t)$ is defined by

$$w(x,t) = \frac{0.1A + 0.5B + C}{A + B + C} \quad \text{and}$$

where $\mathrm{A} = \mathrm{e}^{-0.05(\mathrm{x}-0.5+4.95\mathrm{t})/\nu}, \mathrm{B} = \mathrm{e}^{-0.25(\mathrm{x}-0.5+0.75\mathrm{t})/\nu)}$ and $\mathrm{C} = \mathrm{e}^{-0.5(\mathrm{x}-0.375)/\nu}$.

2. Anisotropic Test Problem. This problem has the interesting feature that the wave moves at right angles to the characteristics. The problem is a slightly modified form of the one used by Zegeling [19] in his work on the Moving Finite Element Method. The p.d.e. is defined by

$$\frac{\partial u}{\partial t} + 3\,u\,\frac{\partial u}{\partial x} + 3\,(1.5-u)\,\frac{\partial u}{\partial y} - 3\,\nu\,(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) = 0\ ,\ \nu = 0.0001\ .$$

The exact solution is

$$u(x,y,t) = \frac{3}{4} - \frac{1}{4 + 4\,e^B}, \quad \text{where } \mathrm{B} = 0.125(-\mathrm{x}+\mathrm{y}-0.75\mathrm{t})/\nu\ .$$

3. Burgers' Test Problem II.

The problem is a nonlinear version of Burgers' problem I .

$$\frac{\partial u}{\partial t} + u\,\frac{\partial u}{\partial x} + u\,\frac{\partial u}{\partial y} - \nu\,(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) = 0, \quad \nu = 0.0001\ ,$$

$$\text{where} \quad (\mathrm{x},\mathrm{y},\mathrm{t}) \ \epsilon\ (0,1) \times (0,1) \times (0.25, 1.25]\ .$$

The exact solution is given by

$$u(x,y,t) = \frac{1}{1 + e^B}\ ,\ \text{where } \mathrm{B} = (\mathrm{x}+\mathrm{y}-\mathrm{t})/\nu.$$

4. Two Dimensional Advection.

This problem consists of the advection of a steep ramp function of width 0.01 and gradient 100 which effectively models a discontinuity.

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0\ ,\ \ (x,y,t)\ \epsilon\ (0,1) \times (0,1) \times (0,1]\ .$$

The analytic solution is given by

$$u(x,y,t) = 1.1 + max\ (\ min\ (\delta, 0)\ ,\ -1)$$

where $\delta = 100\ (0.1 - \frac{1}{2}(x+y) + t)$.

6.1. Testing Procedure. Problems 1-4 were solved using fixed evenly spaced space meshes of 9x9 , 27x27 and 81x81 points. The meshes were constructed so that the mesh points were the centres of square cells and so that the meshes were nested. The Theta method described in Section 3 was used with the functional iteration option switched on. Four different error control strategies were used within the time integration routines:

(i) The standard absolute LEPS strategy used in method of lines codes, that is, controlling the local error, $\underline{le}_{n+1}(t_{n+1})$, with respect to a fixed tolerance, TOL, i.e.

$$|| \ \underline{le}_{n+1}(t_{n+1}) \ || \ \leq \ tol. \tag{64}$$

In this case the stability requirement on functional iteration (19) is not enforced.
(ii) The new strategy presented in Section 4, that is, controlling the local error, $\underline{le}_{n+1}(t_{n+1})$, so that

$$|| \ \underline{le}_{n+1}(t_{n+1}) \ || \ \leq \ \epsilon \ || \ \underline{V}(t_{n+1}) \ - \ \underline{v}(t_{n+1}) \ ||.$$

Although this is a LEPUS strategy it has been implemented in a LEPS form simply by defining *tol* in (64) by

$$tol \ = \ \epsilon \ || \ \underline{V}(t_{n+1}) \ - \ \underline{v}(t_{n+1}) \ ||.$$

In this case again the stability requirement on functional iteration (19) is not enforced.
(iii) The stability timestep control described in Section 3.4 is used with the CFL parameter set to 0.1 or 0.3 except for Problem 2 for which instability occurs with 0.3 and so the values 0.1 and 0.05 are used. The largest CFL value, defined according to (29), that could be used on Problems 1 to 4 was 0.5 , 0.1 , 0.3 and 0.5 respectively as defined by equation (29). In this case again the stability requirement on functional iteration (19) is not enforced.
(iv) Strategy (ii) above is used but with the stability requirement on functional iteration (19) enforced.

In the case of Methods (i) and (ii) above it is important to understand the effect of the error control without an explicit stability condition being present. It is for this reason that the functional iteration stability control is switched off and accuracy alone used to control the timestep. In the case of method (iii) the functional iteration stability requirement has again to be disregarded as it will otherwise interfere with the CFL condition that is specified. In all cases when the functional iteration stability condition is not used two functional iterations are performed per timestep.

6.2. Choice of ϵ . Before comparing the performance of the local error control strategies, we must consider the choice of the parameter ϵ , used in the control strategies (ii) and (iv) above. Lawson et. al. [11] suggest that ϵ should be sufficiently small such that the spatial discretisation error dominates the time integration error but sufficiently large so that the computational work will not be excessive. Numerical testing indicates that ϵ should be in the range 0.1 to 0.5 since for $\epsilon \ > \ 0.5$ the maximum global errors increase sharply. This result is similar to that of Lawson et. al. [11] who suggest using $\epsilon \ = 0.3$ for parabolic problems. This value is found to be optimal for method (iv) but in the case of method (ii), when Method A is used in error estimation and when no stability restriction on functional iteration is enforced, a value of 0.1 is more appropriate. This, in turn, matches results obtained by Berzins [2] for one dimensional hyperbolic p.d.e.s. In the case of method B the error estimates are smaller in size so $\epsilon \ = 0.2$ fulfills the same role. Strategies (ii) and (iv) do not require input from the user, but the user must supply the tolerance to be used in the LEPS control (64) or the CFL number in equation (29). The only way for the user to do this is to be very conservative or to adopt a trial and error approach to find the largest tolerance or CFL number for which the spatial error dominates.

TABLE 1
Results for Problem 1 - Burgers I

		Max. L1 error x 1000 at Time						
NPTS	TOL	T=0.11	T= 0.44	T= 0.77	T= 1.00	NSTEPS	NFCN	CPU
	0.1D-1	34	61	60	55	24	61	0.31
	0.1D-3	25	15	39	43	160	373	1.78
	CFL 0.3	30	15	28	54	36	79	0.42
9x9	CFL 0.1	27	14	29	50	90	187	0.97
	A 0.1	25	17	26	40	212	487	2.47
	B 0.3	25	15	35	46	168	393	2.02
	AS 0.3	25	16	37	50	98	257	1.30
	0.5D-2	19	27	21	20	51	121	4.5
	0.5D-4	11	8.4	20	31	376	865	32.7
	CFL 0.3	11	8.6	23	27	108	223	8.8
27x27	CFL 0.1	10	8.6	20	27	270	547	21.9
	A 0.1	10	8.3	24	30	392	899	34.2
	B 0.2	10	8.6	15	32	336	765	31.0
	AS 0.3	11	8.6	24	31	204	538	20.1
	0.3D-2	12	19	15	15	139	319	109
	0.3D-4	4.5	7.9	5.7	8.2	827	1897	647
	CFL 0.3	4.9	8.3	4.8	7.0	324	655	234
81x81	CFL 0.1	4.6	8.1	4.7	6.9	810	1627	589
	A 0.1	4.6	7.2	5.5	8.9	872	2019	687
	B 0.2	4.6	8.1	4.6	7.0	813	1903	698
	AS 0.3	5.1	6.9	8.5	6.6	432	1117	376

6.3. Comparison of Approaches. Tables 1 to 4 show the results obtained when using the fixed LEPS strategy (6.1) with a range of accuracy tolerances. The CPU time quoted, however, does not reflect the experimentation needed to actually find the best tolerance. From these results it can be seen that by using smaller tolerances the same accuracy is achieved but at a greater cost, while using larger tolerances may increase the efficiency, but larger global errors are obtained, proving that the spatial error is no longer dominating.
Key to Tables 1 to 5.
$NPTS$ is the number of points used in the spatial mesh.
ϵ and TOL are the parameters used in the o.d.e. integration routine.
$MAX\ L1\ ERR$ is the maximum grid error found at the specified output times.
CPU is the amount of CPU time used, measured in seconds.
$NSTEPS$ is the number of time steps used in the integration of the o.d.e.'s.
A 0.1 refers to the use of method A with $\epsilon = 0.1$ and B 0.2 refers to the use of method B with $\epsilon = 0.2$
CFL 0.3 refers to the method of Section 3.4 with $CFL = 0.3$.
AS 0.3 refers to the use of method A with $\epsilon = 0.3$ and the stability requirement (19) enforced.
CS 0.3 refers to the use of method C with $\epsilon = 0.3$ and the stability requirement (19) enforced.

A comparison is now possible with Method C in which the low-order solution is propagated forward in time. Table 4 shows that in all cases the accuracy of the solution is less than if method A or B is used. Note that the functional iteration restriction (19) is used in this case.

6.4. Discussion of the Numerical Results. From Tables 1 to 4 it can be seen that the new local error control strategies yield solutions of comparable accuracy when solving all the problems. Summarising the results, the LEPUS strategy defined by

TABLE 2
Error Results for Anisotropy Problem

		Max. L1 error x 1000 at Time						
NPTS	TOL	T=0.11	T= 0.44	T= 0.77	T= 1.00	NSTEPS	NFCN	CPU
	0.1D-2	16	8.5	8.7	4.9	70	167	0.58
	0.1D-3	18	6.9	8.3	4.9	99	239	0.76
	CFL 0.1	17	6.6	8.3	4.7	90	187	0.64
9x9	CFL .05	17	6.6	8.3	4.7	180	367	1.18
	A 0.1	18	7.1	9.5	4.8	89	217	0.74
	B 0.2	19	6.9	8.5	5.2	74	173	0.59
	AS 0.3	18	6.8	8.3	4.7	130	301	1.0
	0.5D-3	8.0	3.4	4.1	2.3	216	509	13.0
	0.5D-4	6.8	2.6	3.1	1.9	241	573	14.7
	CFL 0.1	6.8	2.4	3.1	1.8	270	547	14.0
27x27	CFL .05	6.8	2.4	3.1	1.8	540	1087	27.7
	A 0.1	6.8	2.5	3.1	1.8	265	617	15.6
	B 0.2	7.0	2.5	3.1	1.8	235	509	13.8
	AS 0.3	7.0	2.4	3.1	1.7	332	758	18.1
	0.3D-3	3.3	2.7	3.2	1.2	677	1605	365
	0.3D-4	2.4	0.9	1.1	0.6	711	1683	377
	CFL 0.1	2.4	0.9	1.1	0.6	810	1627	380
81x81	CFL .05	2.4	0.9	1.1	0.6	1620	3247	758
	A 0.1	2.4	0.9	1.1	0.6	689	1547	351
	B 0.2	2.4	1.0	1.1	0.6	680	1485	366
	AS 0.3	2.4	0.9	1.1	0.6	798	1626	369

TABLE 3
Results for Problem 3. Burgers' II

		Max. L1 error x 1000 at Time						
NPTS	TOL	T=0.26	T= 0.69	T= 1.0	T= 1.3	NSTEPS	NFCN	CPU
	0.1D-1	22	29	37	27	27	67	0.21
	0.1D-2	4.1	25	42	26	46	105	0.32
	CFL 0.3	7.1	24	38	30	36	79	0.27
9x9	CFL 0.1	2.6	23	35	28	90	187	0.58
	A 0.1	2.0	28	47	35	67	190	0.56
	B 0.2	2.0	29	37	27	52	125	0.45
	AS 0.3	5.1	23	39	51	60	138	0.4
	0.5D-2	6.7	10	21	17	82	197	4.7
	0.5D-3	6.9	9	13	9	120	273	6.6
	CFL 0.3	4.7	8.7	13	9.3	108	223	5.5
27x27	CFL 0.1	6.5	9.1	13	9.9	270	547	13.5
	A 0.1	7.0	9.0	12	9.3	146	335	8.2
	B 0.2	7.0	8.1	14	9.5	124	305	7.7
	AS 0.3	4.1	12	13	10	121	327	7.0
	0.3D-2	1.0	26	11	10	243	555	134
	0.3D-3	1.1	3.3	4.3	3.9	311	699	169
	CFL 0.3	0.9	3.1	4.5	3.4	324	655	166
81x81	CFL 0.1	0.9	2.9	4.2	3.2	810	1627	427
	A 0.1	1.1	3.4	4.4	4.1	383	877	211
	B 0.2	1.3	2.5	4.3	4.0	376	873	226
	AS 0.3	1.2	2.8	4.1	3.2	483	997	229

equations (37) and (50) yields, *automatically* , solutions at least as accurate as those obtained when controlling the LEPS with tolerances chosen in order that the spatial discretisation error dominates (as it does for the LEPUS strategies) or when using a CFL stability control alone. The user no longer has to experiment with different accuracy tolerances to find the solution for which the spatial error is dominant.

TABLE 4
First Order Results for Problem 3 - Burgers II

		Max. L1 error x 1000 at Time						
NPTS	TOL	T=0.26	T= 0.69	T= 1.00	T= 1.30	NSTEPS	NFCN	CPU
	0.1D-1	19	48	68	57	32	77	0.2
	0.1D-2	9.5	46	74	58	36	98	0.3
9x9	0.1D-3	2.8	45	67	57	106	237	0.7
	CS 0.3	2.2	47	68	55	60	141	0.4
	0.5D-2	5	19	26	22	84	215	4.5
	0.5D-3	5	18	26	20	104	276	5.8
27x27	0.5D-4	6	18	25	20	210	446	10.2
	CS 0.3	7	18	31	20	86	250	5.1
	0.3D-2	2.0	6.2	8.7	6.7	241	498	114
	0.3D-3	1.6	6.3	8.9	6.8	252	631	131
81x81	0.3D-4	1.5	6.0	8.3	6.4	565	1277	280
	CS 0.3	1.6	6.1	8.3	6.6	300	616	145

TABLE 5
Results for Advection Example - Problem 4

		Max. L1 error x 1000 at Time						
NPTS	TOL	T=0.11	T= 0.55	T= 0.77	T= 1.00	NSTEPS	NFCN	CPU
	0.01	25	110	59	13	33	77	0.24
	0.1D-3	27	82	40	2.0	145	323	1.02
	CFL 0.3	28	86	44	4.7	36	79	0.27
9x9	CFL 0.1	27	82	40	3.8	90	187	0.60
	A 0.1	26	82	37	6.2	131	291	0.94
	B 0.2	30	81	38	5.7	94	207	0.7
	AS 0.3	32	86	42	3.6	68	169	0.52
	0.005	23	65	29	9.5	99	219	5.6
	0.5D-4	14	39	17	0.9	243	513	12.9
	CFL 0.3	14	61	32	1.2	108	223	5.5
27x27	CFL 0.1	13	39	17	0.1	270	547	13.4
	A 0.1	13	39	17	0.8	226	467	11.7
	B 0.2	13	39	16	1.5	262	549	14.6
	AS 0.3	13	39	17	0.1	175	413	9.9
	0.3D-2	10	48	19	4.4	293	697	160
	0.3D-4	4.1	16	6.6	2.3D-2	582	1213	281
	CFL 0.3	5.5	41	18	6.0D-3	324	655	149
81x81	CFL 0.1	4.2	15	6.3	8.3D-5	810	1627	363
	A 0.1	4.2	14	6.1	2.2D-4	962	1921	453
	B 0.2	4.6	15	6.5	1.2D-4	553	1135	280
	AS 0.3	4.5	16	6.9	9.8D-5	383	980	208

In particular the results for method A with $\epsilon = 0.3$ and with a stability condition for functional iteration in place (tabulated as Method AS) appears to be a good automatic time step control combining both accuracy and stability.

The results for Problem 3 illustrate the case when the stability requirement dominates the accuracy requirement. In this case it is difficult to obtain a stable solution which is dominated by temporal error. Nevertheless the implicit stability condition in Methods A, B and AS makes these methods competetive with CFL control on this problem.

One surprising feature is that when the second-order method is used as the primary solution the observed accuracy is only first-order in the $L1$ norm. This happens when a large proportion of the mesh cells (between fifty and ninety percent) use a first-order method rather than the second order method. This has also been observed

in experiments with the adaptive mesh code of Berzins, Lawson and Ware [4]. Part of the reason for this is that many of the problems solved here have solutions with large flat areas and sudden shocks. Although the method A error estimator is only activated in the second order cells the method appears to provide a good means of controlling the temporal error so that the spatial error dominates. A more rigorous approach is to use method B, but it is less clear how to implement the third-order part of this on non-uniform triangular meshes such as those used by Ware and Berzins [18].

7. Conclusions and Further Developments. The aim is to develop a fully automatic general purpose algorithm for the numerical solution of hyperbolic equations using the method of lines. From practical experience, the local error control strategy introduced in Section 4, equation (37), when combined with the stability requirement on functional iteration appears to provide a promising starting point for the development of such an algorithm. This control strategy aims to balance the spatial and temporal errors, although the spatial discretisation error is allowed to dominate so that it can be controlled by remeshing. By computing the accuracy tolerance at each time step the error in the time integration varies in relation to the spatial discretisation error. This reflects the fundamental difference in the method of lines from standard o.d.e. systems principally because there is a spatial discretisation error everpresent. The error control approach taken here reflects this, in contrast to standard local error control or a control based on a CFL number in which experimentation is needed to ensure the spatial error is not corrupted by temporal errors or becomes unstable.

Finally, an important area for further development is in the combination of the error control strategy (51) with the use of adaptive mesh algorithms. The aim to balance the spatial and temporal errors in the method of lines relates directly to the mesh modification techniques which seek to control the spatial discretisation error. A combination of these two algorithms will lead to a method of lines technique which automatically adapts the space mesh in order that the spatial error is controlled and then, accordingly, adjusts the local error tolerance used in the o.d.e. integrator so that the two errors are related in some way. Work is at present underway to achieve this for realistic Euler flows in two space dimensions, see Ware et. al. [18].

REFERENCES

[1] M. Berger and J. Oliger, *Adaptive mesh refinement for hyperbolic partial differential equations* , J. of Comp. Phys., 53 (1984), pp 484–512.

[2] M. Berzins, *Balancing Space and Time Errors for Hyperbolic Equations.* Paper presented at Athens Interdisciplinary Olympia 91, to appear in L.S. Xanthis, editor, Method of Lines and Dimensional Reduction in Computational Mathematics and Mechanics, Kluwer Academic Pub. (to appear 1995).

[3] M. Berzins, and R. M. Furzeland, *An Adaptive Theta Method for the Solution of Stiff and Non-Stiff Differential Equations.* Applied Numerical Mathematics 8 (1992) pp. 1–19.

[4] M. Berzins, J. Lawson and J. Ware, *Spatial and Temporal Error Control in the Adaptive Solution of Systems of Conservations Laws.* Advances in Computer Methods for Partial Differential Equations VII Proc. of 1992 IMACS PDE Conference, IMACS, New Jersey, USA, pp. 60–66 (1992).

[5] L. J.Durlofsky, B. Engquist and S. Osher, *Triangle Based adaptive Stencils for the Solution of Hyperbolic Conservation Laws.* Jour. of Comp. Physics, 54 (1990), pp545–581.

[6] J. Gary, *The Method of Lines Applied to a Simple Hyperbolic Problem,* J. of Comp. Physics, 22 (1976), pp.131–149.

[7] P. H. Gaskell and A. C. Lau, *Curvature Compensated Convective Transport:SMART-A New Boundedness Preserving Algorithm,* Int. J. of Num. Meths. in Fluids, 8 (1988), pp.617-641.

[8] D. J.Higham, *Global Error Versus Tolerance for Explicit Runge-Kutta Methods.* I.M.A. Journal of Numerical Analysis, 11 (1991), pp 457–480.

[9] J. M. Hyman, *A method of lines approach to the numerical solution of conservation laws.* Advances in Computer Methods for Partial Differential Equations III, R. Vichnevetsky and R.S. Stepleman (Eds), IMACS 1979.

[10] B. Koren , *Multigrid and Defect Correction for the Steady Navier-Stokes Equations- Applications to Aerodynamics.* Thesis, Centrum voor Wiskunde en Informatica, Amsterdam, 1989.

[11] J. Lawson, M. Berzins and P.M. Dew , *Balancing Space and Time Errors for Parabolic Equations,* SIAM J. Sci. Stat. Comput., 12 (1991), No 3, pp. 573–594.

[12] J. Lawson and M. Berzins, *Towards an automatic algorithm for the numerical solution of parabolic equations using the method of lines,* Proc. of the 1989 ODE conference. IMA Conference Series (1992), Eds J.R. Cash and I.Gladwell, pp. 309-322.

[13] R. J. Leveque, *Numerical Methods for Conservation Laws,* Lectures in Mathematics Series, Birkhauser Verlag, Basel-Boston-Berlin, (1990).

[14] P. K. Moore and J. E. Flaherty , *Adaptive overlapping grid methods for parabolic systems in two space dimensions,* Jour. of Comp. Physics, 98 (1992), pp54–63.

[15] P. L. Roe , *Finite Volume Methods for the Compressible Navier-Stokes Equations,* pp. 2088-2101 Proceedings of 5th International Conference on Laminar and Turbulent Flow,Montreal 1987, C.Taylor, W.G.Habashi and M.M.Hafez (Eds), Pineridge Press, (1988).

[16] S. Spekreijse , *Multigrid Solution of Monotone Second Order Discretizations of Hyperbolic Conservation Laws,* Math. Comp. 49 (1987) ,No. 179, pp. 135–155.

[17] T. Strouboulis and J. T. Oden, *A Posteriori Error Estimation of Finite Element Approaches in Fluid Mechanics.* Computer Methods in Applied Mechanics and Engineering 78 (1990) pp.201–242.

[18] J. Ware and M. Berzins, *Adaptive Finite Volume Schemes for Fluid-Flow Problems.* pp. 794-798 in Advances in Computer Methods for Partial Differential Equations VII Proc. of 1992 IMACS PDE Conference, IMACS, New Jersey, USA (1992).

[19] P. A. Zegeling, *A Note on the Grid Movement induced by M.F.E. Method.* Report NM-R9019, Centrum voor Wiskunde en Informatica, Amsterdam, (1989).