ADAPTIVE FINITE VOLUME METHODS FOR TIME-DEPENDENT P.D.E.S.

J.WARE AND M.BERZINS *

Abstract. The aim of adaptive methods for time-dependent p.d.e.s is to control the numerical error so that it is less than a user-specified tolerance. This error depends on the spatial discretization method, the spatial mesh, the method of time integration and the timestep. The spatial discretization method and positioning of the spatial mesh points should attempt to ensure that the spatial error is controlled to meet the user's requirements. It is then desirable to integrate the o.d.e. system in time with sufficient accuracy so that the temporal error does not corrupt the spatial accuracy or the reliability of the spatial error estimates. This paper is concerned with the development of a prototype algorithm of this type, based on a cell-centered triangular finite volume scheme, for two space dimensional convection-dominated problems.

Key words. Adaptive Finite-Volume, Time-dependent, P.D.E.s

AMS(MOS) subject classifications. 65M20, 65M15

1. Introduction . Two important trends in numerical methods for the spatial discretization of partial differential equations are the moves towards using unstructured triangular or tetrahedral meshes and using computable error estimates and adaptive algorithms to control the error. While error control forms an important part of many mathematical software algorithms, relatively few attempts have been made to apply error control to convection-dominated problems. This paper describes one such attempt.

Although finite element and finite volumes schemes based on unstructured triangular meshes have been used for many years, only recently have a number of high-order cell-centered finite volume schemes been developed, [9,18,13] . This paper is concerned with one such method, that of Ware and Berzins, [18] , and with how this method may be used as part of a prototype error-control solver for time-dependent systems of conservation laws.

The other main components of this solver are the spatial and temporal error balancing approach of Berzins [1], and the adaptive mesh algorithms of Berzins et al. [4]. This combination of error control strategies not only ensures that the main error present is that due to spatial discretization but offers the tantalising possibility of overall error control.

An outline of this paper is as follows. Section 2 describes the spatial discretization method in detail. A method of lines approach is used to integrate forwards in time in Section 3. The results in Section 4 illustrate the performance of the method on a model problem. Section 5 deals with spatial and temporal error balancing. Section 6 begins an analysis of the properties of the discretization method while the full adaptive algorithm is

* School of Computer Studies, University of Leeds, Leeds LS2 9JT, U.K. 1995

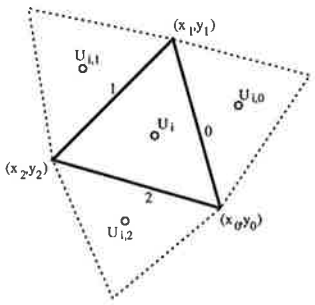

FIG. 2.1. *Discretisation on a triangle*

outlined in Section 7 and applied to a numerical example.

2. Spatial Discretisation Method . The method used in this paper was presented by Ware and Berzins [18,5] based on ideas outlined in Berzins et al [2]. A number of similar triangular mesh cell-centered schemes have appeared recently [7,9,13], which all use similar ideas to those used on structured quadrilateral meshes e.g. [15]. An early scheme was that of Cockburn et al [7] while three other schemes were developed more or less simultaneously [9,13,18] and possess similarities as well as important differences, see Ware, [19], for a comparison. Although the spatial discretization algorithm has been developed for systems of equations, for ease of exposition consider the class of p.d.e.s in cartesian co-ordinates as

$$\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = 0 \tag{2.1}$$

where $f = f(x, y, u)$ and $g = g(x, y, u)$ are the flux functions in x and y respectively and with appropriate boundary and initial conditions.

The cell-centred finite volume scheme described here uses triangular elements as the control volumes over which the divergence theorem is applied. The finite volume representation of a solution is formally piecewise constant within each control volume and is not associated with any particular position. To allow the construction of high order schemes however the centroid of the triangle is defined as the nodal position and the solution value is associated with that point. In Figure 2.1 for example, the solution at the centroid of triangle i is U_i and the solutions at the centroids of the triangles surrounding triangle i are $U_{i,0}$, $U_{i,1}$ and $U_{i,2}$.

Integration of equation (2.1) on the ith triangle gives:

$$\int_{A_i} \frac{\partial u}{\partial t} d\Omega = -\int_{A_i} \left(\frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} \right) d\Omega, \tag{2.2}$$

where A_i is the area of triangle i. The area integral on the left hand side of equation (2.2) is approximated by a one point quadrature rule.

The quadrature point is the centroid of triangle i. By using the divergence theorem the area integral on the right hand side is replaced by a line integral around the triangular element :

$$A_i \frac{\partial U_i}{\partial t} = - \oint_{C_i} (f.\underline{n}_x g.\underline{n}_y) dS, \tag{2.3}$$

where C_i is the circumference of triangle i. The line integral along each edge is approximated by using the midpoint quadrature rule. The numerical flux is evaluated at the midpoint of and normal to the edge :

$$\begin{aligned} \frac{\partial u}{\partial t} = -\frac{1}{A_i}(& f_{i,0}\Delta y_{0,1} - g_{i,0}\Delta x_{0,1} + f_{i,1}\Delta y_{1,2} \\ & -g_{i,1}\Delta x_{1,2} + f_{i,2}\Delta y_{2,0} - g_{i,0}\Delta x_{2,0}), \end{aligned} \tag{2.4}$$

where $\Delta x_{i,j} = x_j - x_i$ and $\Delta y_{i,j} = y_j - y_i$. The fluxes $f_{i,j}$ and $g_{i,j}$ are evaluated by using (approximate) Riemann solvers f_{Rm} and g_{Rm} respectively. At the midpoint of each edge one-dimensional Riemann problems are solved in the cartesian directions with the *internal* and *external* values being used as the initial conditions:

$$\begin{aligned} \frac{\partial u}{\partial t} = -\frac{1}{A_i}(& f_{Rm}(U^-_{i,0}, U^+_{i,0})\Delta y_{0,1} - g_{Rm}(U^-_{i,0}, U^+_{i,0})\Delta x_{0,1} + \\ & f_{Rm}(U^-_{i,1}, U^+_{i,1})\Delta y_{1,2} - g_{Rm}(U^-_{i,1}, U^+_{i,1})\Delta x_{1,2} + \\ & f_{Rm}(U^-_{i,2}, U^+_{i,2})\Delta y_{2,0} - g_{Rm}(U^-_{i,2}, U^+_{i,2})\Delta x_{2,0}), \end{aligned} \tag{2.5}$$

where $U^-_{i,j}$ is the internal solution, with respect to triangle i, on edge j and $U^+_{i,j}$ is the external solution, with respect to triangle i, on edge j.

A standard first-order scheme uses the piecewise constant solution on either side of the edge as the upwind values, $U^-_{i,j} = U_i$, $U^+_{i,j} = U_{i,j}$, but introduces excessive numerical diffusion in the solution.

2.1. Limited Interpolants in Two Dimensions. Spekreijse [15] derived a higher order scheme on quadrilateral meshes by using linear upwind values to create left and right values for the Riemann solver. These upwind linear values were limited to ensure undershoot and overshoot could not occur. This approach will now be used on unstructured meshes in which case the internal and external values at cell interface of two triangular elements, $U^-_{i,j}$ and $U^+_{i,j}$ in equation (2.5) are replaced with the limited linearly interpolated values defined by

$$U^-_{i,j} = U_i + \Phi(r^-_{i,j}) \left(U^{-L}_{i,j} - U_i\right), \tag{2.6}$$

$$U^+_{i,j} = U_{i,j} + \Phi(r^+_{i,j}) \left(U^{+L}_{i,j} - U_{i,j}\right), \tag{2.7}$$

where $U^{-L}_{i,j}$ is the internal linear upwind value , $U^{+L}_{i,j}$ is the external linear upwind value, $r^-_{i,j}$ is the internal upwind bias ratio of gradients and $r^+_{i,j}$ is the external upwind bias ratio of gradients. The internal and external

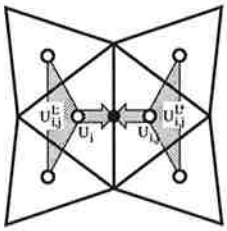

FIG. 2.2. *Linear Upwind Values*

ratio of linear gradients are defined in a similar manner to that used by Spekreijse [15] by

$$r^-_{i,j} = \frac{U^C_{i,j} - U_i}{U^{-L}_{i,j} - U_i} \text{ and } r^+_{i,j} = \frac{U^C_{i,j} - U_{i,j}}{U^{+L}_{i,j} - U_{i,j}}. \tag{2.8}$$

where $U^C_{i,j}$ is the linear centred value at the cell interface. The function $\Phi(.)$ is a limiter function such as the Van Leer limiter often used in one-dimensional schemes e.g.

$$\Phi(r^-_{i,j}) = \frac{r^-_{i,j} + |r^-_{i,j}|}{1 + |r^-_{i,j}|}. \tag{2.9}$$

The choice of this function is discussed further in Section 6 below. Equations (2.6), (2.7) and (2.8) describe the unstructured flux limiter scheme but in terms of new, and as yet undefined, interpolated and extrapolated values: $U^{-L}_{i,j}$, $U^{+L}_{i,j}$ and $U^C_{i,j}$. The upwind value $U^{-L}_{i,j}$ is constructed by forming a linear interpolant using the solution values at three centroids. The first centroid is the centroid of triangle i which has value U_i associated with it. The other two are the centroids of the neighbouring triangles on edges other than edge j of triangle i, see Figure 2.2. The other upwind value $U^{+L}_{i,j}$ is defined in a similar way. For certain meshes the three centroid points may be collinear in which case it is not possible to define a linear interpolant. In this case the immediate upwind centroid value will be used: internally U_i or externally $U_{i,j}$.

The centered value, $U^C_{i,j}$, is constructed from the six values: U^A, U^B, U^C, U^D, U^E and U^F (see Figure 2.3) by a series of one-dimensional linear interpolations. Three linear interpolations are performed using *opposing* pairs of centroid values, see Figure 2.3. U^G, U^I and U^H are found using the pairs U^D and U^E, U^A and U^B and U^C and U^F respectively. If the midpoint of the edge lies between U^H and U^I, as in Figure 2.3, then the centred value is found by linear interpolation using these two values. Otherwise the values U^G and U^I are used to linearly interpolate the centred value at the midpoint.

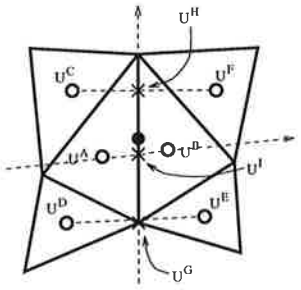

FIG. 2.3. *Linear Centred Value*

2.2. Boundary Conditions . As the spatial discretisation scheme traverses several spatial elements in order to construct the linear interpolants care must be taken if one or more of the elements is not present due to a boundary.

In the case of Dirichlet conditions the specified solution value at the boundary is used as the *external* one in the Riemann solver. The linear centred value $U_{i,j}^{C}$ is now formed by averaging the internal solution, U_i, and the boundary condition value. The internal upwind linear $U_{i,j}^{-L}$ is constructed as normal. This allows the internal limited linear value $U_{i,j}^{-}$ to be constructed as usual. Dirichlet boundary conditions can also be used in constructing the upwind linear interpolant for $U_{i,j}^{-L}$ or $U_{i,j}^{+L}$. Should one of the triangles in Figure 2.2 be a boundary edge then the boundary value provided can be used instead. The boundary value is associated with the midpoint of the edge rather than a centroid.

The derivative information provided by the Neumann condition can be used to estimate the external solution value by using the the internal solution value and the derivative provided by the Neumann condition. A *ghost* cell is created outside the boundary that is the reflection of the triangular element lying on the boundary. Using the internal centroid value U_i at the midpoint of the edge and the derivative normal to the edge the centroid value in the ghost cell is extrapolated. This extrapolated value in the ghost cell is used as the external solution value. The same procedure is used as for the Dirichlet boundary condition.

3. Time Integration. The above spatial discretization scheme results in a system of differential equations, each of which is of the form of equation (2.5) . This system of equations can be written as the initial value problem:

$$\dot{\underline{U}} \;=\; \underline{F}_N \;(\; t,\; \underline{U}(t)\;) \;\;,\; \underline{U}(0) \;\; \textit{given}\;, \tag{3.1}$$

where the N dimensional vector, $\underline{U}(t)$, is defined by

$$\underline{U}(t) \;=\; (U(x_1,y_1,t)\;,\; U(x_2,y_2,t),...,U(x_N,y_N,t)\;)^T\;.$$

The point (x_i, y_i) is the centroid of the ith triangle and $U(x_i, y_i, t)$ is a numerical approximation to $u(x_i, y_i, t)$. Numerical integration of (3.1) provides the approximation, $\underline{V}(t)$, to the vector of exact p.d.e. solution values at the mesh points, $\underline{u}(t)$. The global error in the numerical solution can be expressed as the sum of the spatial discretization error, $\underline{e}(t) = \underline{u}(t) - \underline{U}(t)$, and the global time error, $\underline{g}(t) = \underline{U}(t) - \underline{V}(t)$. That is,

$$\begin{aligned} \underline{E}(t) = \underline{u}(t) - \underline{V}(t) &= (\underline{u}(t) - \underline{U}(t)) + (\underline{U}(t) - \underline{V}(t)) \\ &= \underline{e}(t) + \underline{g}(t). \end{aligned} \tag{3.2}$$

The Theta method code of Berzins and Furzeland [3] used here selects functional iteration automatically for the non-stiff o.d.e.s resulting from the Euler equations. The numerical solution at $t_{n+1} = t_n + k$ where k is the time step size, as denoted by $\underline{V}(t_{n+1})$ is defined by

$$\underline{V}(t_{n+1}) = \underline{V}(t_n) + (1-\theta)k\ \dot{\underline{V}}(t_n) + \theta\ k\ \underline{F}_N(t_{n+1}, \underline{V}(t_{n+1})) \ , \tag{3.3}$$

in which $\underline{V}(t_n)$ and $\dot{\underline{V}}(t_n)$ are the numerical solution and its time derivative at the previous time t_n and the default value of θ is 0.55 . The system of equations is solved using functional iteration, see [1] ,

$$\underline{V}(t_{n+1}^{m+1}) = \underline{V}(t_n) + (1-\theta)k\ \dot{\underline{V}}(t_n) + \theta\ k\ \underline{F}_N(t_{n+1}, \underline{V}(t_{n+1}^{m})), \tag{3.4}$$

where $m = 0, 1, ...$ with a predictor specified by [3].

This method also has an interesting similarity to the iterated Leap-Frog method of Hyman, [11], as defined by

$$\begin{aligned} \underline{V}(t_{n+1}) = [\ \underline{V}(t_n)(2 + 3r - r^3) &+ r^3\ Vu_{n-1}\ (1+r)^2\ k\ \dot{\underline{V}}(t_n) \\ &+(1+r)\ k\ \underline{F}_N(t_{n+1}, \underline{V}(t_{n+1}))\ ]/(2+3r), \end{aligned} \tag{3.5}$$

where the ratio $r = k/k_{n-1}$ and $k_{n-1} = t_n - t_{n-1}$. After some manipulation this may be rewritten as

$$\begin{aligned} \underline{V}(t_{n+1}) = \underline{V}(t_n) + \frac{1+r}{2+3r}\ k\ \underline{F}_N(t_{n+1}, \underline{V}(t_{n+1})) + \frac{1+2r}{2+3r}\ k\ \dot{\underline{V}}(t_n) \\ + \frac{r^3}{2+3r}\ [\ \underline{V}(t_{n-1})\ - \underline{V}(t_n)\ +\ k_{n-1}\ \dot{\underline{V}}(t_n)\ ]. \end{aligned} \tag{3.6}$$

Substituting $\theta = \frac{1+r}{2+3r}$ enables this to be again rewritten as

$$\begin{aligned} \underline{V}(t_{n+1}) = \underline{V}(t_n) + (1-\theta)k\ \dot{\underline{V}}(t_n) + \theta\ k\ \underline{F}_N(t_{n+1}, \underline{V}(t_{n+1})) \\ +(0.5-\theta)2r^2\ [\ \underline{V}(t_{n-1}) - \underline{V}(t_n)\ +\ k_{n-1}\ \dot{\underline{V}}(t_n)\ ] \end{aligned} \tag{3.7}$$

which is the theta method with a local extrapolation type correction term. In most time dependent p.d.e. codes either a CFL stability control is employed or a standard o.d.e. solver is used which controls the local error

$\underline{l}_{n+1}(t_{n+1})$ using local time error per step, (LEPS), control with respect to a user supplied accuracy tolerance, TOL, i.e.

$$|| \ \underline{l}_{n+1}(t_{n+1}) \ || < TOL. \tag{3.8}$$

or TOL is multiplied by the timestep k in a local time error per unit step (LEPUS) control. When controlling the LEPS it is difficult to establish a relationship between the accuracy tolerance, TOL , and the global time and space errors. In contrast, if the LEPUS is controlled then it is well known that the time global error is proportional to the tolerance. This suggests the use of LEPUS error control with a tolerance TOL that reflects the spatial error present.

4. Numerical Example. This section will demonstrate the effectiveness of the new scheme both in terms of the accuracy achieved and the absence of undershoots and overshoots on the two-dimensional Burger's equation,

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left[\frac{u^2}{2}\right] + \frac{\partial}{\partial y}\left[\frac{u^2}{2}\right] = 0, \tag{4.1}$$

with initial conditions $g(x, y)$. This equation is solved in a circular domain of radius one centered at the origin with a zero Dirichlet boundary condition. The Theta method of Section 3 is used to integrate the o.d.e. system. An absolute and relative tolerance of 10^{-5} is used for the temporal integration. The mesh contains 8192 triangular elements. The Engquist-Osher approximate Riemann solver for Burger's equation is used. The solution profile is shown in Figure 4.1 and is obtained by sampling the solution along the line from $(-1.1, -1.1001)$ to $(1.1001, 1.0)$. The values plotted are the centroid values of the triangles through which this line passes. The crosses are the results using the higher order scheme and the diamonds are the results using the first-order scheme.

The initial conditions were chosen to be a square wave in the radial direction with circular symmetry so that the discontinuity is not aligned with triangle element edges. This provides a good test of how well the scheme will capture discontinuities in general. The Figure 4.1 shows that the new higher order scheme captures the discontinuous profile far better than the first-order scheme.

5. Spatial and Temporal Errors. Efficient time integration requires that the spatial and temporal are roughly the same order of magnitude. The need for spatial error estimates unpolluted by temporal error requires that the spatial error is the larger of the two errors. Lawson and Berzins, [12] have developed a strategy for parabolic equations which achieves this by controlling the local time error to be a fraction of the growth in the spatial discretization error over a timestep. Berzins, [1], describes a similar strategy for hyperbolic equations, based on the local growth in the spatial

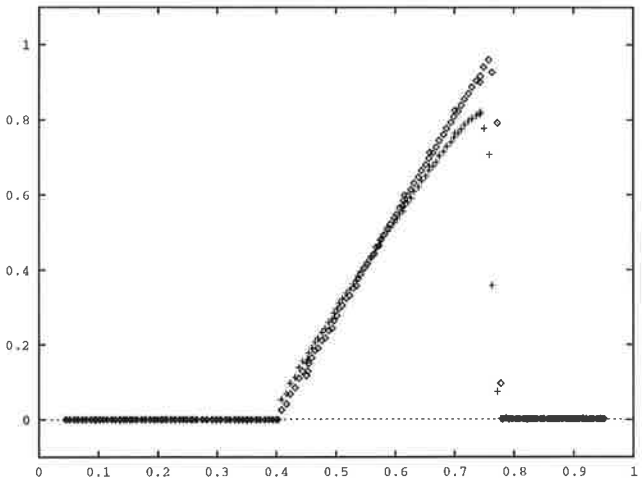

FIG. 4.1. *Burger's equation (time = 0.8).*

error over each timestep. The local-in-time spatial error, $\underline{\hat{e}}(t_{n+1})$, for the timestep from t_n to t_{n+1} is defined as the spatial error at time t_{n+1} given the assumption that the spatial error, $\underline{e}(t_n)$, at time t_n is zero. A local-in-time error balancing approach is then given by

$$|| \ \underline{l}_{n+1}(t_{n+1}) \ || \ < \ \epsilon \ || \ \underline{\hat{e}}(t_{n+1}) \ ||, \quad 0 < \epsilon < 1. \tag{5.1}$$

The error $\underline{\hat{e}}(t_{n+1})$ is estimated by the difference between the computed second-order in space solution and the first-order piecewise constant solution which satisfies a modified o.d.e. system denoted by

$$\dot{\underline{v}}_{n+1} \ (t) = \underline{G}_N(t, \underline{v}_{n+1}(t)), \tag{5.2}$$

where $\underline{v}_{n+1}(t_n) = \underline{V}(t_n)$, $\dot{\underline{v}}_{n+1}(t_n) = \underline{G}_N(t, \underline{V}(t_n))$ and where $\underline{G}_N(.,.)$ is obtained from $\underline{F}_N(.,.)$ simply by setting the limiter function in the space discretisation to zero. The local-in-time space error is then given by

$$\underline{\hat{e}}(t_{n+1}) = \underline{V}(t_{n+1}) - \underline{v}_{n+1}(t_{n+1}) \tag{5.3}$$

and is computed by applying the θ method of the previous section to equation (5.2) . As only an estimate of the error is required it is sufficient to use only one functional iteration to compute $\underline{v}_{n+1}$; combining this with the conditions on $\underline{v}_{n+1}(t_n)$ gives

$$\begin{aligned} \underline{v}_{n+1}(t_{n+1}) \ = \ \underline{V}(t_n) \ + \ \theta \ k \ \underline{G}_N(t_{n+1}, \underline{V}(t_{n+1})) \ + \\ (1-\theta) \ k \ \underline{G}_N(t_n, \underline{V}(t_n)). \end{aligned} \tag{5.4}$$

Combining equation (5.4) with equations (3.3) and (5.3) gives

$$\begin{aligned} \underline{\hat{e}}(t_{n+1}) = \theta \quad k \quad & [\underline{F}_N(t_{n+1}, \underline{V}(t_{n+1})) - \underline{G}_N(t_{n+1}, \underline{V}(t_{n+1}))] \; + \\ (5.5) \qquad (1-\theta) \quad k \quad & [\underline{F}_N(t_n, \underline{V}(t_n)) - \underline{G}_N(t_n, \underline{V}(t_n))] \; . \end{aligned}$$

This estimate will only approximate the error in the low-order solution and so local extrapolation in space is effectively being used when the high-order solution is used to move forward in time. The actual situation is a little more complicated due to the fact that the high order scheme switches to first-order at shocks and when the solution is flat and consequently may use the high order limiter only for a portion of the grid, see [1].

As the right side of equation (5.5) has a factor of k the error control (5.1) for the step to t_{n+1} is of the LEPUS form given by

$$(5.6) \quad || \; \underline{l}_{n+1}(t_{n+1}) \; || \; < \; k \; TOL \quad where \; TOL = \epsilon \; || \; \underline{\hat{e}}(t_{n+1})/k \; ||.$$

Although LEPUS control is generally thought to be inefficient for standard o.d.e.s, equation (5.1) may be also used directly as a LEPS control, with little difference in integration performance.

6. Analysis of Discretization Method. Recent work by Struijs, Deconinck and Roe [16] proposes two new properties that two or three space dimensional schemes should posess.The concepts of *second order accuracy* and *monotonicity preservation* are generalised to *linearity preservation* and *positivity* . The definition of positivity, [16], requires that every new value U_i^{n+1} can be written as a convex combination of old values:

$$(6.1) \qquad U_i^{n+1} = \sum_{j=0}^{ntri} c_j U_j^n \quad \text{with} \quad \forall c_j \geq 0$$

while $\sum c_j = 1$ for consistency. This guarantees ,[16], a maximum principle for the discrete steady state solution thus prohibiting the occurrence of new extrema and imposing stability on the explicit scheme.

Consider the linear advection equation

$$(6.2) \qquad \frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} = 0,$$

where a and b are positive constants. Using the scheme presented earlier, see equation (2.4) and Figure 2.1, the discrete form is

$$\begin{aligned} (6.3) \frac{\partial u}{\partial t} = -\frac{1}{A_i}[\; & f_{Rm}(U_{i,0}^-, U_{i,0}^+)\Delta y_{0,1} \; - \; g_{Rm}(U_{i,0}^-, U_{i,0}^+)\Delta x_{0,1} \; + \\ & f_{Rm}(U_{i,1}^-, U_{i,1}^+)\Delta y_{1,2} \; - \; g_{Rm}(U_{i,1}^-, U_{i,1}^+)\Delta x_{1,2} \; + \\ & f_{Rm}(U_{i,2}^-, U_{i,2}^+)\Delta y_{2,0} \; - \; g_{Rm}(U_{i,2}^-, U_{i,2}^+)\Delta x_{2,0} \;]. \end{aligned}$$

The Riemann problems for the linear advection are simple to solve. The solution is the product of the upwind value and the constant.

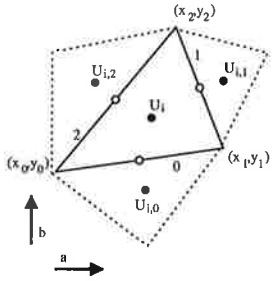

FIG. 6.1. *A possible triangle to characteristic alignment.*

Assume that the triangle is aligned to the characteristic directions as in Figure 6.1. The discrete form then simplifies to

$$\begin{aligned}\frac{\partial U_i}{\partial t} = -\frac{1}{A_i}[\ & (aU^-_{i,0})\Delta y_{0,1} - (bU^+_{i,0})\Delta x_{0,1} + \\ & (aU^-_{i,1})\Delta y_{1,2} - (bU^-_{i,1})\Delta x_{1,2} + \\ & (aU^+_{i,2})\Delta y_{2,0} - (bU^-_{i,2})\Delta x_{2,0}\ ].\end{aligned} \tag{6.4}$$

From equations (2.6) and (2.7) it can be seen that these internal and external values at the cell interface are a combination of the centroid values and linear upwind values. In the case when the limiting function $\Phi()$ is zero the centroid solution values are used at the cell interface and the equation may be expressed as:

$$\begin{aligned}\frac{\partial U_i}{\partial t} &= \frac{1}{A_i}\left[-a\Delta y_{0,1} - a\Delta y_{1,2} + b\Delta x_{1,2} + b\Delta x_{2,0}\right] U_i \\ &+ \frac{1}{A_i}[\ b\Delta x_{0,1}\ U_{i,0} - \ a\Delta y_{2,0}\ U_{i,2}]\,.\end{aligned} \tag{6.5}$$

All the internal centroid solution values have $\Delta x_{i,j} < 0$ and $\Delta y_{i,j} > 0$. and all the external centroid solution values have $\Delta x_{i,j} > 0$ and $\Delta y_{i,j} < 0$. The factors of the external centroid solution values are all positive and the factors of all the internal centroid solution values are all negative. Note the $\Delta x_{i,j}$ and $\Delta y_{i,j}$ go anticlockwise around the triangular element so

$$\Delta x_{0,1} + \Delta x_{1,2} + \Delta x_{2,0} = \Delta y_{0,1} + \Delta y_{1,2} + \Delta y_{2,0} = 0. \tag{6.6}$$

So far only the spatial discretisation has been considered. To obtain the final result the temporal discretisation will have to be introduced. Applying the forward Euler method to equation (6.5) gives

$$\begin{aligned}U_i^{n+1} = U_i^n &+ \frac{k_n}{A_i}\left[-a\Delta y_{0,1} - a\Delta y_{1,2} + b\Delta x_{1,2} + b\Delta x_{2,0}\right] U_i^n \\ &+ \frac{k_n}{A_i}[\ b\Delta x_{0,1}\ U^n_{i,0} - a\Delta y_{2,0} U^n_{i,2}]\,.\end{aligned} \tag{6.7}$$

This shows that the contribution from the external centroid solution values $U_{i,0}$ and $U_{i,2}$ is positive. By summing all the factors of U_i the following inequality must be satisfied if the scheme is to be positive:

$$1 + \frac{k_n}{A_i}[-a\Delta y_{0,1} - a\Delta y_{1,2} + b\Delta x_{1,2} + b\Delta x_{2,0}] \geq 0. \tag{6.8}$$

This can be simplified using equation (6.6) and rewritten as,

$$\frac{k_n}{A_i}[-a\Delta y_{2,0} + b\Delta x_{0,1}] \leq 1, \tag{6.9}$$

where $-\Delta y_{2,0} \geq 0$ and $\Delta x_{0,1} \geq 0$. If L_i is the length of the longest edge of the triangle i then $-\Delta y_{2,0}, \Delta x_{0,1} \leq L_i$ and a sufficient condition for positivity is,

$$k_n \frac{L_i}{A_i}(a+b) \leq 1. \tag{6.10}$$

This is a CFL type stability condition that depends on the term L_i/A_i which is also used as a measure of the *quality* of a triangle. So providing the inequality in equation (6.10) is satisfied the discretisation can be written in the form of equation (6.1) with all the $c_j \geq 0$. Although only one possible alignment to the characteristic directions has been considered similar results are produced by considering the other possibilities.

In the case when the limiter $\Phi(.)$ is non zero, Berzins and Ware [6] considered different flow paths through the triangle in Figure 6.1 and showed that three sufficient conditions for positivity are:

1. For every upwind interpolant the centroid value nearest the edge at whose midpoint the upwind value is being calculated is the maximum or minimum of the three values used to form the interpolant.

2. The centred interpolant must be ~~bounded~~ by the centroid values ~~on either side~~ i.e.

~~$$U^C_{i,0} = \alpha U_{i,0} + (1-\alpha)U_i \ , \quad \text{for } 0 \leq \alpha \leq 1. \tag{6.11}$$~~

3. The limiter $\Phi(.)$ must be positive and $\Phi(S)/S \leq 1$. This last condition is satisfied, for example, by a modified van Leer limiter defined by

$$\Phi(S) = \frac{S + |S|}{1 + v} \quad \text{where} \quad v = \text{Max}(1, |S|) \ . \tag{6.12}$$

Although the standard Van Leer limiter has been used for the experiments described in this paper Berzins and Ware [6] showed that this limiter may give very slight overshoots and undershoots.

A linearity-preserving spatial discretization method is defined by [16], as one which preserves the exact steady state solution whenever this is a linear function of the space coordinates x and y, for any arbitrary triangulation of the domain. This is equivalent to second order accuracy on regular

meshes, see [16]. Berzins and Ware were able to show that the method in its unlimited form is linearity preserving but that in some cases condition 2 above may force linearity preservation to be violated.

7. Prototype Automatic Software. A prototype package which combines the spatial and temporal discretization methods and associated error control strategies described above has been written. The package requires the user to provide an approximate Riemann solver for the convective fluxes, definitions of the source and diffusive terms and the boundary and initial conditions. In addition the user must provide a file containing the spatial domain description for the mesh generator and a spatial error tolerance for the adaptivity software. An optional user-supplied monitoring function is called at the end of every successful timestep.

The goal of the automatic algorithm described here is to ensure that the spatial mesh is fine or coarse enough so that the solution satisfies the users' accuracy and efficiency requirements. The adaptive algorithm was developed from that in Berzins et. al. [2], using the Shell Research mesh generator based on the ideas of George et al. [10] , and adopting a regular subdivision approach , see Berzins et al. [4]. The strategies for deciding when to remesh are essentially those of Lawson and Berzins [12]. At each time step the estimate of $||\underline{\hat{e}}(t)||$ is calculated, and if it is greater than 0.25 of EPS - the user supplied tolerance, then a new mesh is constructed that ensures that the subsequent error is less than a given fraction of EPS. The underlying assumption in this agorithm is that the introduction of extra mesh points will cause the error to decrease. The selection of appropriate remeshing times is made by using a combination of present estimated errors and predicted future errors. Once a new mesh has been found a "flying restart "is used. The computed solution and the time history array used by the time integrator are interpolated using the method of Ramshaw [14] onto this mesh and the time integration is restarted with the same time step as used immediately before remeshing. Care must be taken to modify the accuracy tolerance for the time integration so that it reflects the expected reduction in the spatial discretization error. An illustration of how the solver works is provided by the dimensional Burgers' equation problem defined by :

$$(7.1) \quad u_t + u\,u_x + u\,u_y - \nu\,(u_{xx} + u_{yy}) = 0, \quad \nu = 0.0001\ ,$$

where $(x,y,t)\ \epsilon\ (0,1) \times (0,1) \times (0.25,1.25]$. The exact solution is given by $u(x,y,t) = (1 + e^B)^{-1}$, where $B = (x+y-t)/(2\nu)$.

Three runs were done, the first (FX) used a fixed mesh of 8192 triangles and a time local error TOL of 1.0e-5 in an L1 vector norm. The second (AD) used the adaptive space strategy with a space local error control of 1.0d-5 and the same ordinary local error control and the third (AU) used the fully automatic code with error balancing and adaptivity. The cpu times are those on an Silicon Graphics R4000. The "NT" rows show the

number of triangles used by the adaptive codes. The adaptive algorithm provides the same accuracy using less triangles and c.p.u. time than the fixed mesh code. "NR" is the number of spatial remeshes automatically selected. Similar results are given by Berzins and Ware [5] using an earlier version of the code.

TABLE 7.1
Adaptive Mesh Burgers' Equation Results.

	L1 Error Norm at Time							
	0.26	0.69	1.0	1.3	NS	NF	CPU	NR
FX	4.0D-3	3.9D-2	5.2D-2	2.1D-2	745	1769	2828	0
AD	1.1D-2	2.9D-2	7.2D-2	1.0D-2	1805	4664	1490	229
NT	902	1173	1223	330				
AU	3.8D-3	2.5D-2	5.3D-2	1.2D-2	5625	15446	1636	666
NT	508	710	290	210				

8. Summary. This paper has presented a new spatial discretisation scheme for unstructured meshes that is an extension of one-dimensional flux limiter schemes. The numerical experiments show that this new scheme has better capture than the first-order scheme without undershoots or overshoots. The prototype adaptive software based on this discretisation has been used to solve a variety of convection-dominated problems using fully automatic mesh generation and mesh adaptation algorithms. The adaptivity tracks features in the solution automatically whilst using large elements away from these features to increase the efficiency. The package also can be used on a variety of different computer architectures. The flux calculation used in the residual is designed to operate in parallel allowing the package to take advantage of both the shared memory parallel architecture of Silicon Graphics machines and distributed memory parallel architectures [17].

Acknowledgements The authors would like to thank Shell Research Ltd and SERC for funding via a CASE award for JW.

REFERENCES

[1] M. BERZINS.,*Temporal error control for convection-dominated equations in two spaced dimensions.*, SIAM Journal of Scientific Computing (to appear), 199x.

[2] M. BERZINS, P. L. BAEHMANN, J. E. FLAHERTY, AND J. LAWSON., *Towards an automated finite element solver for time-dependent fluid-flow problems.* In The Mathematics of Finite Elements and Application VII, pages 181–188. Academic Press, 1991.

[3] M. BERZINS AND R.M. FURZELAND. *An adaptive theta method for the solution of stiff and non-stiff differential equations.* App. Num. Math., 8:1–19,1992.

[4] M, BERZINS, J.M. WARE AND J. LAWSON. *Spatial and Temporal error Control in the Adaptive Solution of Systems of Conservation Laws.* Advances in Com-

puter Methods for Partial Differential Equations: IMACS PDE VII. IMACS, 1992.

[5] M. Berzins and J.M. Ware. *Reliable finite volume methods for time-dependent p.d.e.s.* In J.R. Whiteman, editor, Mafelap Conference. John Wiley, 1993.

[6] M. Berzins and J.M. Ware. *Positive discretization methods for hyperbolic equations on irregular meshes.* Submitted to Applied Numerical Mathematics, 1994.

[7] B. Cockburn, Suchung Hou, and Chi-Wang Shu. *The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case.* Math. of Comp., 54(190):545–581, 1990.

[8] B. Cockburn and Chi-Wang Shu. *Tvb runge-kutta local projection discontinuous galerkin finite element method for conservation laws II: General framework.* Math. of Comp., 52(186):411–435, 1989.

[9] L. J. Durlofsky, B. Enquist, and S. Osher. *Triangle based adaptive stencils for the solution of hyperbolic conservation laws.* Jour.Of Comp. Phys., 98:64–73, 1992.

[10] P. L. George, F. Hecht, and E. Saltel. *Automatic mesh generator with specified boundary.* Comp. Meths. in Appl. Mech. and Eng., 92:269–288, 1991.

[11] J.M. Hyman. *A method of lines approach to the numerical solution of conservation laws.* In Advances in Comp. Meths. for P.D.E.s III, IMACS, 1979.

[12] J.L. Lawson and M. Berzins. *Towards an automatic algorithm for the numerical solution of parabolic p.d.e.s using the method of lines.* In J.R. Cash and I. Gladwell, editors, Computational Ordinary Differential Equations, pages 309–322. Oxford University Press, 1992.

[13] S.Y. Lin, T.M. Wu, and Y.S. Chin. *Upwind finite-volume method with a triangular mesh for conservation laws.* Jour. of Comp. Phys., 107:324–337, 1993.

[14] J.D. Ramshaw. *Conservative rezoning algorithm for generalized two-dimensional meshes.* Jour. of Comp. Phys., 59:193–199, 1985.

[15] S. Spekreijse. *Multigrid solution of monotone second-order discretisations of hyperbolic conservation laws.* Math. of Comp., 49(179):135–155, 1987.

[16] R. Struijs, H. Deconinck, and P. L. Roe. *Fluctuation splitting schemes for the 2D Euler equations.* Technical report, von Karman Institute for Fluid Dynamics, Chaussee de Waterloo, 72, B-1640 Rhode Saint Genese - Belgium, 1991.

[17] C.H. Walshaw and M. Berzins. *Enhanced dynamic load-balancing of adaptive unstructured meshes.* In R.F. Sincovec et. al., editor, Parallel Processing for Scientific Computing, pages 971–978. SIAM, 1993.

[18] J.M. Ware and M. Berzins. *Finite volume techniques for time-dependent fluid-flow problems.* In Advances in Computer Methods for Partial Differential Equations: IMACS PDE VII. IMACS, 1992.

[19] J.M. Ware. *The Adaptive Solution of Time-Dependent PDEs in Two Space Dimensions.* Ph.D. Thesis, School of Computer Studies, The University of Leeds.