# Reliable Finite Volume Methods for Time-Dependent Partial Differential Equations

M. Berzins and J. Ware

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

## 20.1 INTRODUCTION

The accuracy of numerical solutions to time-dependent partial differential equations (p.d.e.s) depends on the spatial discretization method, the spatial mesh, the method of time integration and the timestep. In particular the spatial discretization method and positioning of the spatial mesh points should ensure that the spatial error is controlled to meet the user's requirements. It is then desirable to integrate the ordinary differential equation (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 describes the prototype of such an automated solver which has been developed from the framework laid down by Berzins *et al.*(1991). The components of this solver are the spatial and temporal error balancing approach of Berzins (1993), the discretization method of Ware and Berzins (1993) and the adaptive mesh algorithms of Berzins *et al.*(1993). 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.
The approach used in the solver is described as follows. Section 2 describes the problem class and spatial discretization method. A method of lines approach is used which allows the global error to be decomposed into spatial and temporal errors in Section 3 and the spatio-temporal error balancing strategy to be described in Section 4. The performance of this strategy is illustrated in Section 5 while the full adaptive method is outlined in Section 6 and applied to a numerical Euler equations example.

## 20.2 FINITE VOLUME DISCRETIZATION METHOD

For ease of exposition consider the class of p.d.e.s in cartesian co-ordinates as

$$u_t = ( f(u, u_x, u_y) )_x + ( g(u, u_x, u_y) )_y \ , t \in (0, t_e] \ , \ (x, y) \in \Omega \qquad (20.1)$$

with appropriate boundary and initial conditions.

*The Mathematics of Finite Elements and Applications*
Edited by J. R. Whiteman 

The spatial discretization algorithm has been developed for systems of equations of the form of (20.1). The algorithm uses upwind methods developed for the Euler equations for the convective parts of the fluxes $f$ and $g$ and uses centered discretization methods for the diffusive parts of the fluxes. Such discretizations are considered by many authors, e.g. Spekreise (1987) computes solutions free of oscillations using quadrilateral meshes. Ware and Berzins (1993) and Durlofsky *et al.*(1991) extend the methods to the use of unstructured triangular meshes. A finite volume approach is adopted to integrate equations (20.1) over a triangular element $i$. Solution values are assumed to be located at the centroids of the triangles. Applying the divergence theorem and a one point quadrature rule along the edges of the triangle gives

$$Area_i \dot{u}_i = -\sum_{j=1}^{3} [(f(u_{i,j}, (u_{i,j})_x, (u_{i,j})_y)\Delta y_j - (g(u_{i,j}, (u_{i,j})_x, (u_{i,j})_y)\Delta x_j \ ] \tag{20.2}$$

where $u_{i,j}$ is the solution value midway along the edge $j$ , $\Delta x_j$ is the change in the $x$ co-ordinate in going from one end of the edge to the other , $u_i$ is the solution value associated with the centroid of the $i$ th triangle and the other values in the equation are similarly defined.

A standard first-order scheme uses only piecewise constant values inside each triangle; the evaluation of the convective fluxes midway along the edge involves the approximate solution of three one-dimensional Riemann problems in the direction of the normals to the edges of the triangle. This is done by using the standard scheme of Osher with a van Leer limiter.

The extension to a second order scheme on an unstructured triangular mesh is a cell-centered finite volume approach that uses a six triangle stencil around each edge, Ware and Berzins (1993), giving a ten triangle stencil for the discretization of the convective terms on each triangle.

Calculation of the derivative values used in the diffusive flux evaluations at the mid-points of an edge is by a control volume approach using the two triangles on either side of the edge. The divergence theorem is applied and mid-point quadrature used to evaluate the integrals of the solution along the edges of the two-triangle control volume. The mid-point solution values are calculated using linear interpolation based on the centroid values of the two triangle control volume plus the centroid value of one of the triangles external to the control volume. This interpolant has the advantege that it uses values pre-calculated for the convective part of the discretization. The derivative values at the mid-point of an edge thus involve contributions from six centroid solution values. Calculation of all the edge derivatives for a triangle thus involves the same ten point stencil as is used for the convective terms.

## 20.3 TIME INTEGRATION

The above spatial discretization scheme results in a system of differential equations, each of which is of the form of equation (20.2) . This system of equations can be

written as the initial value problem:

$$\dot{U} = F_N\ (\ t,\ U(t)\ )\ ,\ U(0)\ \ given\ , \tag{20.3}$$

where the $N$ dimensional vector, $U(t)$, is defined by

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

The point $(x_i, y_i)$ is the centroid of the $i$th triangle and $U(x_i, y_i, t)$ is a numerical approximation to $u(x_i, y_i, t)$ . Numerical integration of (20.3) provides the approximation, $V(t)$, to the vector of exact p.d.e. solution values at the mesh points, $u(t)$ . The global error in the numerical solution can be expressed as the sum of the spatial discretization error, $e(t) = u(t) - U(t)$, and the global time error, $g(t) = U(t) - V(t)$. That is,

$$\begin{aligned} E(t) = u(t) - V(t) &= (u(t) - U(t)) + (U(t) - V(t)) \\ &= e(t) + g(t). \end{aligned} \tag{20.4}$$

The Theta method code of Berzins and Furzeland (1992) 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 $V(t_{n+1})$ is defined by

$$V(t_{n+1}) = V(t_n) + (1-\theta)k\ \dot{V}(t_n) + \theta\ k\ F_N(t_{n+1}, V(t_{n+1}))\ , \tag{20.5}$$

in which $V(t_n)$ and $\dot{V}(t_n)$ are the numerical solution and its time derivative at the previous time $t_n$ and the default value of $\theta$ is 0.55 .
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 $l_{n+1}(t_{n+1})$ using local time error per step, (LEPS), control with respect to a user supplied accuracy tolerance, $TOL$, i.e.

$$||\ l_{n+1}(t_{n+1})\ ||\ <\ TOL. \tag{20.6}$$

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.

## 20.4 BALANCING THE SPACE AND TIME ERRORS

Efficient time integration requires that the spatial and temporal errors 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 (1992) have developed a strategy for parabolic equations which achieves this by controling the local time error to be a fraction of the growth in the spatial discretization error over a timestep. Berzins (1993) describes a similar strategy for

hyperbolic equations, based on the local growth in the spatial error over each timestep. The local-in-time spatial error, $\hat{e}(t_{n+1})$, 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, $e(t_n)$ , at time $t_n$ is zero. A local-in-time error balancing approach is then given by

$$|| \ l_{n+1}(t_{n+1}) \ || < \epsilon || \ \hat{e}(t_{n+1}) \ ||, \ 0 < \epsilon < 1. \tag{20.7}$$

The error $\hat{e}(t_{n+1})$ 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

$$\dot{v}_{n+1}(t) = G_N(t, v_{n+1}(t)), \tag{20.8}$$

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

$$\hat{e}(t_{n+1}) = V(t_{n+1}) - v_{n+1}(t_{n+1}) \tag{20.9}$$

and is computed by applying the $\theta$ method of the previous section to equation (20.8). As only a rough estimate of the error is required it is sufficient to use only one functional iteration to compute $v_{n+1}$; combining this with the conditions on $v_{n+1}(t_n)$ gives

$$v_{n+1}(t_{n+1}) = V(t_n) + \theta k G_N(t_{n+1}, V(t_{n+1})) + (1-\theta) k G_N(t_n, V(t_n)). \tag{20.10}$$

(It is also possible to ignore the $(1-\theta)$ term as in Berzins *et al.*(1992).) Combining equation (20.10) with equations (20.5) and (20.9) gives

$$\begin{aligned} \hat{e}(t_{n+1}). \ = \ & \theta k[F_N(t_{n+1}, V(t_{n+1})) - G_N(t_{n+1}, V(t_{n+1}))] \ + \\ & (1-\theta) k[F_N(t_n, V(t_n)) - G_N(t_n, V(t_n))] \ . \end{aligned} \tag{20.11}$$

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. As the right side of equation (20.11) has a factor of $k$ the error control (20.7) for the step to $t_{n+1}$ is of the LEPUS form given by

$$|| \ l_{n+1}(t_{n+1}) \ || < k \ TOL \ \text{ where } TOL = \epsilon \ || \ \hat{e}(t_{n+1})/k \ ||. \tag{20.12}$$

Berzins (1993) gives an analysis to show that although the method attempts to control accuracy a Courant-like stability condition is also satisfied. Although LEPUS control is generally thought to be inefficient for standard o.d.e.s, equation (20.7) may be also used directly as a LEPS control, with little difference in integration performance. The estimate of the spatial error used improves on that suggested by Berzins *et al.*(1991) using $h$-extrapolation as the present approach estimates the error in each triangle individually rather than on groups of four triangles merged into one large triangle.

## 20.5 NUMERICAL EXPERIMENTS

Consider the two dimensional Burgers' equation problem defined by :

$$u_t + u\,u_x + u\,u_y - \nu\,(u_{xx} + u_{yy}) = 0, \quad \nu = 0.0001\,, \tag{20.13}$$

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

The problem was solved using fixed evenly-spaced square space meshes of 9x9 and 81x81 points and a cell-centered discretization used by Berzins (1993). The time integration was performed using the LEPUS strategy given by equation (20.12) with $\epsilon = 0.3$ and $0.5$ , and with the standard absolute LEPS strategy given by equation (20.6) .

In Table 20.1 "NP" is the number of spatial mesh points. "TOL" is the time integration "LEPS" tolerance except that "New 0.5" refers to the new strategy, with $\epsilon = 0.5$ . "L1 ERR" is the spatial error in the L1 norm at the specified output times. "CPU" is the time used on an Iris Indigo R4000. "NS" is the number of time steps used in the integration of the o.d.e.s. "NF" is the number of o.d.e. function evaluations and "EST 0.3" is the norm of the local-in-time space error estimator with $\epsilon = 0.3$.

Table 20.1 shows that as the local error tolerance is decreased the spatial error dominates. It can also be seen that the error balancing strategy with $\epsilon = 0.3$ or with $\epsilon = 0.5$ provides temporal integration results that are dominated by the spatial error and are computed efficiently. The Table also shows that the local-in-time spatial error estimates mimic the actual errors in Table 20.1 . A number of experiments using the new error control strategy are described by Berzins (1993).

**Table 20.1** Fixed Mesh Burgers' Equation Results.

| NP | TOL | L1 Error Norm at Time | | | | NS | NF | CPU |
|---|---|---|---|---|---|---|---|---|
| | | 0.26 | 0.69 | 1.0 | 1.3 | | | |
| | 0.5D-1 | 1.1D-2 | 2.4D-2 | 3.5D-2 | 2.8D-2 | 32 | 91 | 0.2 |
| 9 | 0.5D-2 | 1.1D-2 | 2.4D-2 | 3.5D-2 | 2.8D-2 | 48 | 130 | 0.4 |
| by | 0.5D-3 | 2.6D-2 | 2.3D-2 | 3.3D-2 | 2.8D-2 | 73 | 189 | 0.6 |
| 9 | New 0.3 | 5.5D-3 | 2.2D-2 | 3.3D-2 | 2.6D-2 | 60 | 138 | 0.4 |
| | New 0.5 | 5.5D-3 | 2.2D-2 | 3.4D-2 | 2.6D-2 | 50 | 133 | 0.3 |
| | Est 0.3 | 5.0D-3 | 4.0D-3 | 3.3D-3 | 4.9D-3 | | | |
| | 0.1D-1 | 2.4D-3 | 4.5D-3 | 5.9D-3 | 3.9D-3 | 244 | 711 | 142 |
| 81 | 0.1D-2 | 1.2D-3 | 2.8D-3 | 4.2D-3 | 3.1D-3 | 452 | 966 | 219 |
| by | 0.1D-3 | 9.5D-4 | 2.9D-3 | 4.0D-3 | 3.1D-3 | 538 | 1175 | 325 |
| 81 | New 0.3 | 1.1D-3 | 2.8D-3 | 3.9D-3 | 3.0D-3 | 483 | 997 | 229 |
| | New 0.5 | 1.1D-3 | 2.8D-3 | 3.9D-3 | 3.0D-3 | 468 | 982 | 225 |
| | Est 0.3 | 4.7D-4 | 6.4D-4 | 8.7D-4 | 6.8D-4 | | | |

## 20.6 A PROTOTYPE AUTOMATIC ALGORITHM

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.*(1991), using the Shell Research mesh generator based on the ideas of George *et al.*(1991), and adopting a regular subdivision approach (e.g. Lohner (1987)), see Berzins *et al.*(1993). The strategies for deciding when to remesh are essentially those of Lawson and Berzins (1992). The input required from the user consists only of the problem specification, an initial spatial mesh from the mesh generator. and an error tolerance for the spatial discretization error, $EPS$. At each time step the estimate of $||\hat{e}(t)||$ is calculated, and if it is greater than 0.25 of $EPS$, then a new mesh is constructed that ensures that the subsequent error is less than $EPSDN$ where $EPSDN$ is a 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 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 Burgers' equation problem used above. The results are shown in Table 20.2.

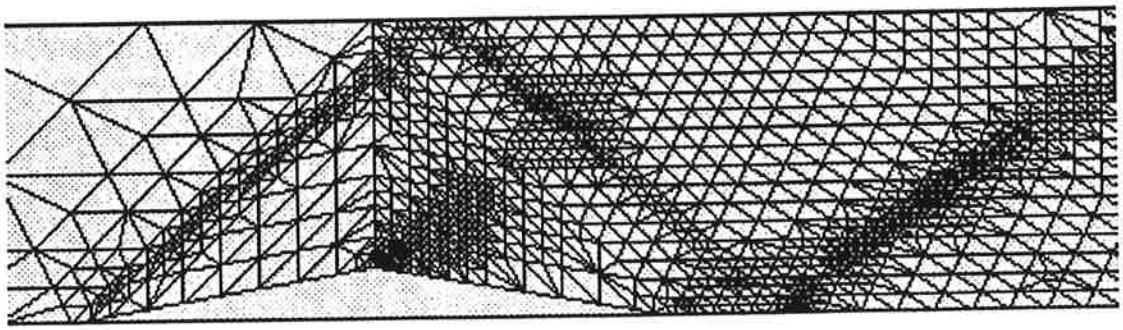

**Figure 20.1** Final Mesh for Euler Equations Example.

**Table 20.2** Adaptive Mesh Burgers' Equation Results.

| TYPE | L1 Error Norm at Time | | | | NS | NF | CPU | NRM |
|---|---|---|---|---|---|---|---|---|
| | 0.26 | 0.69 | 1.0 | 1.3 | | | | |
| FIXD | 1.0D-2 | 4.0D-2 | 3.8D-2 | 2.3D-2 | 720 | 1831 | 3602 | 0 |
| ADAP | 1.9D-2 | 4.0D-2 | 3.8D-2 | 2.4D-2 | 1458 | 3758 | 1361 | 160 |
| NTRI | 970 | 1378 | 931 | 251 | | | | |
| AUTO | 1.9D-2 | 3.1D-2 | 2.6D-2 | 3.2D-2 | 1613 | 4156 | 1009 | 194 |
| NTRI | 762 | 856 | 322 | 217 | | | | |

Three runs were done, the first (FIXD) used a fixed mesh of 8192 triangles and a time local error $TOL$ of 1.0e-5 in an L1 vector norm. The second (ADAP) 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 (AUTO) used the fully automatic code with error balancing and adaptivity. The cpu times are those on an Silicon Graphics R4000. The "NTRI" rows show the number of triangles used by the adaptive codes. The adaptive algorithm provides the same accuracy using less triangles and cpu. time than the fixed mesh code. "NRM" is the number of spatial remeshes automatically selected.
The final example is a well known Euler equations example of Mach 2 supersonic flow down a channel with a small triangular bump. The problem is solved as a time dependent problem by using initial conditions that suppose the bump is missing. The adaptive code solves the problem using 194 meshes ranging between 84 and 2899 triangles. A portion of the final triangulation is shown in Figure 20.1 with the reflecting shock patterns clearly illustrated by the position of the refined triangles.

## 20.7 CONCLUSIONS

This paper describes a prototype solver for time dependent p.d.e.s . The results obtained from preliminary experiments indicate that the algorithm is a promising start to developing codes which automatically control the error in the computed solution. The numerical results shows that this can be achieved for both simple convection problems and for Euler flows in two space dimensions.

## ACKNOWLEDGEMENT

Justin Ware acknowledges the funding by SERC and Shell Research Ltd through a CASE studentship.

## REFERENCES

Berzins M. (1993), Temporal Error Control in the Method of Lines for Convection Dominated Equations. *School of Computer Studies Report 93.18*, to appear in *SIAM J. Sci. Comp.*

Berzins M. , Baehmann P. , Flaherty J.E. and Lawson J. (1991), Towards Reliable Software for Time-Dependent Problems in CFD. *The Mathematics of Finite Elements and Applications VII* Mafelap 1990, pp 181-88. Ed J.R. Whiteman, Academic Press, London.

Berzins M. and Furzeland R.M. (1992), An Adaptive Theta Method for the Solution of Stiff and Non-Stiff Differential Equations . *Applied Num. Math.*, **7**: 1–19.

Berzins M. , Lawson J. and Ware J. (1993), Spatial and Temporal Error Control in the Adaptive Solution of Systems of Conservations Laws. *Advances in Computer Methods for Partial Differential Equations VII* , pp. 60–66 Proc. of 1992 Conf., IMACS, New Jersey, USA.

Durlofsky L.J. , Engquist B. and Osher S. (1990), Triangle Based adaptive Stencils for the Solution of Hyperbolic Conservation Laws. *J. Comp. Physics* , **54**: 545–581.

Georges P.L. Hecht F. and Saltel E. (1991), Automatic Mesh Generation with Specified Boundary. *Comp. Meth. Applied Mech. Eng.* , **92**: 269–288.

Lawson J.L. and Berzins M. (1992), Towards an Automatic Algorithm for the Numerical Solution of Parabolic P.D.E.s using the Method of Lines. *Computational Ordinary Differential Equations* , pp 309–322. Eds. J.R. Cash and I.Gladwell, IMA Conference Series 1992, Oxford University Press.

Lohner R.L. (1987), An adaptive finite element scheme for transient problems in CFD. *Comp. Meths. in Applied Mech. and Eng.*, **61**: 323–338.

Spekreise S. (1987), Multigrid Solution of Monotone Second Order Discretizations of Hyperbolic Conservation Laws. *Mathematics of Computation* , **49**: 135–155.

Ware J. and Berzins M. (1993), Finite Volume Techniques for Time-Dependent Fluid-Flow Problems. *Advances in Computer Methods for Partial Differential Equations VII* , pp. 794–798 Proc. of 1992 Conf., IMACS, New Jersey, USA.