SOLVING CONVECTION AND CONVECTION REACTION PROBLEMS USING THE M.O.L. [⋆]

M. Berzins and J.M. Ware. [1]

Centre for CFD and School of Computer Studies, The University of Leeds, Leeds LS2 9JT, UK.

Abstract

The application of software based on the method of lines (m.o.l.) to convection and convection-reaction problems is considered. The components of a prototype software package, SPRINT2D, are unstructured mesh finite volume algorithms, coupled spatial and temporal error control algorithms and modified time integration techniques for the large systems of o.d.e.s. The software is applied to a realistic combustion problem and the lessons learned used to discuss the validity of the m.o.l. approach.

1 Introduction.

The method of lines for solving time dependent p.d.e.s may be described as 'discretize in space and then integrate in time', usually using o.d.e. initial value software. The two main components of a method of lines package are the spatial discretization method and the time integration code. In this paper we shall consider the application of one such algorithm to a number of problems including a realistic combustion problem.

Many of the practical engineering problems being solved are defined on irregular solution domains. One popular approach for dealing with such problems is to use discretization methods based on unstructured triangular or tetrahedral meshes. One such method, that of Berzins and Ware [5], will be described briefly in Section 2 of this paper. Once spatial discretization has been performed it is necessary to integrate the resulting o.d.e. system forward in time. This raises two important issues. The first issue is that when implicit methods are being used to solve problems with stiff chemistry source terms it is

[⋆] Expanded version of a talk presented at the mol workshop Lexington, Kentucky
[1] supported by Shell Research Ltd and EPSRC

necessary to solve very large systems of linear equations at each time step. This issue will be addressed in Section 3 by introducing an operator splitting approach in the nonlinear equations solver. The second important issue is how the temporal error should be controlled given that there is a spatial error present. This is particularly important when the spatial discretization error is controlled by the use of one of the many adaptive algorithms available, e.g. see Strouboulis and Oden [19] or Moore and Flaherty [11]. 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 dealt with in Section 4 by using the approach of Berzins [2], to balance the spatial and temporal errors. This balancing approach requires an estimate of how the spatial error grows in time. The effectivity of such an estimate is shown in Section 5 using numerical examples. The final section of this paper describes how all these components are brought together in solving a realistic combustion problem. A brief comparison is given between the m.o.l. approach and a more traditional c.f.d. approach . The m.o.l. approach with temporal error control is found to be more reliable but more computationally expensive.

2 Finite Volume Spatial Discretization Methods.

Finite volume methods are commonly used for the semi-discretization of fluid flow equations. These methods are the basis for many different schemes suitable for convection dominated problems. Examples of the types of methods often applied to regular meshes are given by [15,2] and in the references therein. For problems with very general spatial geometries one important approach is to consider methods which are based on unstructured triangular or tetrahedral meshes. Although finite element and finite volume schemes based on unstructured triangular meshes have been used for many years, only recently, see [5], have a number of high-order cell-centered finite volume schemes been developed for convection dominated problems.

The discretization method used in this work, [5], has been developed for systems of equations, but for ease of exposition will be described by considering the class of scalar p.d.e.s:

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x} f(x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}) + \frac{\partial}{\partial y} g(x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}) = h(x, y, u) \tag{1}$$

where f and g are the flux functions in x and y respectively, the source term is h and appropriate boundary and initial conditions are supplied. The cell-centered finite volume scheme described here uses triangular elements as the control volumes over which the divergence theorem is applied. The solution values are deemed to be associated with the centroids of the triangles. In Figure 1, for example, the solution at the centroid of triangle i is U_i, the solutions at the centroids of the triangles surrounding triangle i are U_l, U_j and U_k and the next level of centroid values used by the discretization method on the ith triangle are: U_m, U_n, U_p, U_q, U_r and U_s. The mesh point at which a solution value, say U_s, is defined is denoted by (x_s, y_s) . Integration of equation (1) on the ith triangle, which has area A_i, and use of the divergence theorem gives:

$$A_i \frac{\partial U_i}{\partial t} \;=\; A_i \; h(x_i, y_i, U_i) \;-\; \oint_{C_i} \; (f.\underline{n}_x \;+\; g.\underline{n}_y) \; dS$$

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

$$\begin{aligned} \frac{\partial U_i}{\partial t} = \quad & h(x_i, y_i, U_i) - \frac{1}{A_i}[f_{ik}\Delta y_{0,1} - g_{ik}\Delta x_{0,1} + f_{ij}\Delta y_{1,2} \\ & -g_{ij}\Delta x_{1,2} + f_{il}\Delta y_{2,0} - g_{il}\Delta x_{2,0}] \end{aligned} \tag{2}$$

where $\Delta x_{i,j} = x_j - x_i$, $\Delta y_{i,j} = y_j - y_i$. The fluxes f_{ij} and g_{ij} in the x and y directions respectively are evaluated at the midpoint of the triangle edge separating the triangles associated with U_i and U_j. The convective parts of these fluxes are evaluated by using approximate Riemann solvers f_{Rm} and g_{Rm} respectively with the *left* solution value being defined as that internal to triangle i and the *right* solution value being defined as that external to triangle i:

$$\begin{aligned} \frac{\partial U_i}{\partial t} \;=\; & h(x_i, y_i, U_i) \;-\; \frac{1}{A_i}[f_{Rm}(U^l_{ik}, U^r_{ik}, (U_{ik})_x, (U_{ik})_y)\Delta y_{0,1} - \\ & g_{Rm}(U^l_{ik}, U^r_{ik}, (U_{ik})_x, (U_{ik})_y)\Delta x_{0,1} + f_{Rm}(U^l_{ij}, U^r_{ij}, (U_{ij})_x, (U_{ij})_y)\Delta y_{1,2} - \\ & g_{Rm}(U^l_{ij}, U^r_{ij}, (U_{ij})_x, (U_{ij})_y)\Delta x_{1,2} + f_{Rm}(U^l_{il}, U^r_{il}, (U_{il})_x, (U_{il})_y)\Delta y_{2,0} - \\ & g_{Rm}(U^l_{il}, U^r_{il}, (U_{il})_x, (U_{il})_y)\Delta x_{2,0})], \end{aligned} \tag{3}$$

where U^l_{ij} is the internal solution, with respect to triangle i, at the midpoint of the edge between U_i and U_j and U^r_{ij} is the external solution, with respect to triangle i, on edge j. Note that $U^r_{i,j} = U^l_{j,i}$ as a consequence of this notation. Standard approximate Riemann solvers such as those of Osher [14] and Roe [16] are used to define the convective fluxes. The left and right values for the Riemann solver U^l_{ij} and U^r_{ij} in equation (3) are the limited linearly interpolated values defined by

$$U^l_{ij} = U_i + \Phi(r^l_{ij}) \left(U^L_{ij} - U_i\right) \quad \text{and} \quad \mathrm{U^r_{ij} = U_j + \Phi(r^r_{ij}) \left(U^R_{ij} - U_j\right)}, \tag{4}$$

where U_{ij}^L is the internal linear upwind value, U_{ij}^R is the external linear upwind value, r_{ij}^l is the internal upwind bias ratio of gradients and r_{ij}^r is the external upwind bias ratio of gradients. The internal and external ratio of linear gradients are defined by

$$r_{ij}^l = \frac{U_{ij}^C - U_i}{U_{ij}^L - U_i} \quad \text{and} \quad \mathrm{r_{ij}^r} = \frac{\mathrm{U_{ij}^C - U_j}}{\mathrm{U_{ij}^R - U_j}}. \tag{5}$$

U_{ij}^C is the linear centered value at the cell interface. The limiter function $\Phi(.)$ is described below, but when set to zero gives a first-order method. Equations (4) and (5) depend on the as yet undefined, interpolated and extrapolated values: U_{ij}^L, U_{ij}^R and U_{ij}^C.

The value U_{ij}^L is constructed by using linear extrapolation based on the solution value U_i and an intermediate solution value (again calculated by linear interpolation) U_{lk} which lies on the line joining the centroids at which U_l and U_k are defined (see Figure 1 and [5]). The value U_{ij}^R is defined in a similar way using linear extrapolation based on the solution value U_j and an intermediate solution value (itself calculated by linear interpolation) U_{rs} which lies on the line joining the centroids at which U_r and U_s are defined, see Figure 1. In the case when the three centroid points are collinear it is not possible to define a linear interpolant and so the immediate upwind centroid value will be used: internally U_i or externally U_j.

The centered value, U_{ij}^C, is constructed from the six values: U_i, U_j, U_k, U_l, U_s and U_r by a series of one-dimensional linear interpolations. Three linear interpolations onto the edge being considered are performed using *opposing* pairs of centroid values, see Figure 1. U_{lr}, U_{ij} and U_{ks} are found using the pairs U_l and U_r, U_i and U_j and U_k and U_s respectively. If the midpoint of the edge lies between U_{ks} and U_{ij} then the centered value is found by linear interpolation using these two values. Otherwise the values U_{lr} and U_{ij} are used to compute the centered value at the midpoint by using linear interpolation. An extension of this approach is also used to compute the diffusive flux contributions at mid-points of edges, see [4].

Two important properties of this scheme are those of *linearity preservation* and *positivity*, as proposed by Struijs et. al., [19], see [5]. The definition of positivity requires that every value at a particular time can be written as a convex combination of values at the previous time step. Berzins and Ware [5] showed that the sufficient conditions for positivity are that the limiter $\Phi(.)$ must be positive, $\Phi(S)/S \leq 1$ and some minor restrictions must be placed on the spatial mesh. The limiter condition is satisfied, for example, by a modified

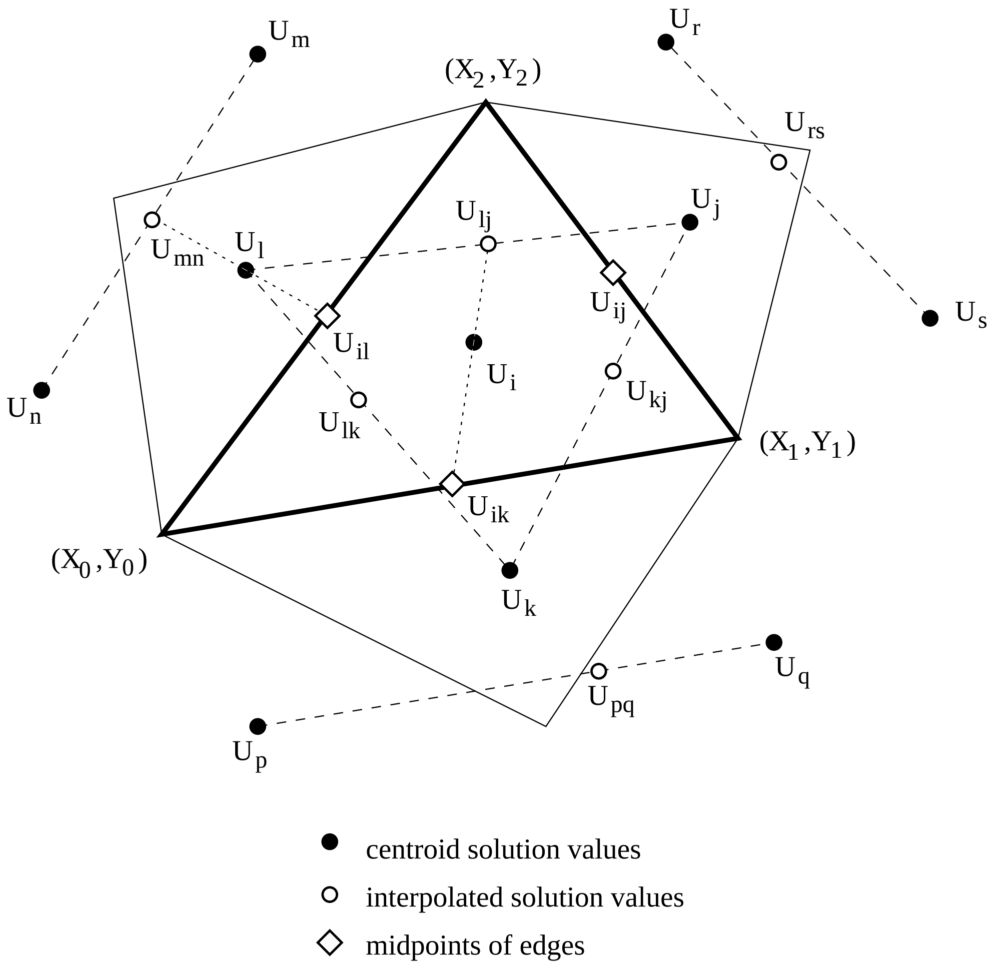

Fig. 1. Construction of Interpolants

van Leer limiter defined by

$$\Phi(S) = \frac{S + |S|}{1 + \mathrm{Max}(1, |\mathrm{S}|)} \quad . \tag{6}$$

A linearity-preserving method is 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,[19].

3 Time Integration.

The above spatial discretization scheme results in a system of differential equations, which can be written as the initial value problem:

$$\dot{\underline{U}} \;=\; \underline{F}_N \;\; (\, t, \; \underline{U}(t) \,) \;\; , \; \underline{U}(0) \;\; \text{given} \; , \tag{7}$$

where the vector, $\underline{U}(t)$, is defined by $\underline{U}(t) \;=\; [U(x_1, y_1, t), ..., U(x_N, y_N, t)\;]^T$. The point x_i, y_i is the centre of the i th cell and $U_i(t)$ is a numerical approximation to the exact solution to the p.d.e. evaluated at the centroid i.e. $u(x_i, y_i, t)$. A method of lines approach is used to numerically integrate equation (7) thus generating an approximation, $\underline{V}(t)$, to the vector of exact p.d.e. solution values at the mesh points, $\underline{u}(t)$. At present two time integration methods are used: Theta method with functional and/or Newton iteration or the backward differentiation formula method using Newton Krylov methods [17]. Both codes allow automatic control of the local error. The Theta method code, see [2,3], which has been used for the experiments described here, defines the numerical solution at $t_{n+1} = t_n + k$, where k is the time step size, as denoted by $\underline{V}(t_{n+1})$, by

$$\underline{V}(t_{n+1}) \;=\; \underline{V}(t_n) \;+\; (1-\theta)k\;\dot{\underline{V}}(t_n) + \theta\;k\;\underline{F}_N(t_{n+1}, \underline{V}(t_{n+1})), \qquad (8)$$

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 $\theta = 0.55$. This system of equations is solved by either using functional iteration or a Newton Krylov method or by automatically switching between the two methods, [3]. Berzins and Ware [5] show that the method will preserve positivity if a CFL-like condition is satisfied and that this in turn is the case if functional iteration converges quickly.

In the case when a Newton-Krylov method is used the equations to be solved for the correction to the solution $\underline{\Delta V}$ for the $p+1$ th iteration of the modified Newton iteration used with the Theta method are:

$$[I - k\theta J]\;\underline{\Delta V} \;=\; \underline{r}\,(t^p_{n+1}) \qquad (9)$$

where

$$\underline{r}\,(t^p_{n+1}) \;=\; -\,\underline{V}(t^p_{n+1}) + \underline{V}(t_n) + (1-\theta)k\dot{\underline{V}}(t_n) - \theta k\underline{F}_N(t_{n+1}, \underline{V}(t^p_{n+1})),$$

$$J \;= \frac{\partial\;\underline{F}_N}{\partial \underline{U}}\;, \quad \text{and} \quad \underline{\Delta \mathrm{V}} \;=\; \left[\underline{\mathrm{V}}(\mathrm{t}^{\mathrm{p+1}}_{\mathrm{n+1}}) - \underline{\mathrm{V}}(\mathrm{t}^{\mathrm{p}}_{\mathrm{n+1}})\right]. \qquad (10)$$

The solution of this system of equations constitutes the major computational task of a method of lines calculation. In cases where large o.d.e. systems that result from the discretization of the flow problems with complex chemistry (50,000 equations typically), the c.p.u. times seem to be excessive, even when using Newton-Krylov methods to solve the nonlinear equations.

One approach which overcomes this is to use a form of operator splitting based on a decomposition of the p.d.e. s into a set of flow terms and a source reactive term. A number of such splitting methods are described by Chung [7]. Consider the o.d.e. function $\underline{F}_N\ (t,\ \underline{U}(t)\ )$ defined by equation (9) and decompose it into two parts:

$$\underline{F}_N\ (t,\ \underline{U}(t)\ )\ =\ \underline{F}_N^f\ (t,\ \underline{U}(t)\ )\ +\ \underline{F}_N^s\ (t,\ \underline{U}(t)\ ) \tag{11}$$

where $\underline{F}_N^f\ (t,\ \underline{U}(t)\ )$ represents the discretization of the convective flux terms f and g in equation (1) and $\underline{F}_N^s\ (t,\ \underline{U}(t)\ )$ represents the discretization of the of the source term h in the same equation. A standard splitting approach, [7], is to employ the following approximation to the Jacobian matrix used by the Theta method within a Newton iteration:

$$I - k\theta J\ \approx\ [I - k\theta\ J_f]\quad [I - k\theta\ J_s\ ]\ )\ +\ O(k^2). \tag{12}$$

where

$$J_f\ =\ \frac{\partial\ \underline{F}_N^f}{\partial \underline{U}}\ \ ,\ \ J_s\ =\ \frac{\partial \underline{F}_N^s}{\partial \underline{U}}\ .$$

The disadvantage of this is that it introduces a second-order splitting error. Fortunately this error only alters the rate of convergence of the iteration as the residual being reduced is still that of the full o.d.e. system. The matrix $I - k\theta J_s$ is the Jacobian of the discretization of the time derivatives and the source terms in the vector $\underline{h}$. This matrix is thus block-diagonal with as many block as there are triangles and with each block having as many rows and columns as there are p.d.e.s. The fact that the block relate only to the chemistry within each cell means that each block may be inverted (or the equations may be solved) independently using LU decomposition. Although for more complex chemical reaction terms it is necessary to adopt the latter approach for the combustion problem considered in Section 6 it is easy to analytically invert the matrix. Consider the five o.d.e.s that represent the discretization of a these source terms on a single triangle. For this triangle the relevant part of the matrix $I - k\theta J_s$ has the particularly simple form of the identity matrix with only one row of off-diagonal non-zero elements. This row consists of the partial derivatives of the p.d.e. source term with respect to the solution variables corresponding to that triangle. The remaining part of the Jacobian $I - k\theta J_f$ has only flow equation terms from the Euler equations. As the spatial discretization method connects each triangle to as many as ten others it follows that the matrix $[I - k\theta\ J_f]$ may have ten times as many entries as the matrix $[I - k\theta \ J_s \ ]$. Approximating the matrix $[I - k\theta \ J_f]$ by the identity matrix as is done by [3] thus eliminates many of the full Jacobian entries.

The new operator-splitting iteration may thus be written as

$$\underline{\Delta V^*} \quad = \quad [I - k\theta J_s]^{-1} \ \ \underline{r}\,(t^p_{n+1}) \tag{13}$$

where $\underline{\Delta V^*}$ is the operator splitting approximation to $\underline{\Delta V}$. The advantage of this is that functional iteration may be used to solve the nonlinear equations provided that the residuals on each triangle are multiplied by the inverse of the matrix $I - k\theta J_s$. This modified form of functional iteration has been implemented as a new linear algebra module inside SPRINT2D software and has been found to give increases in speed-up by a factor of between five and ten over using a Newton-Krylov method for the combustion problem described below.

3.2 Operator Splitting Error.

The disadvantage of introducing operator splitting is that it is difficult to evaluate the error resulting from this. As the splitting is only used to speed up the solution of the nonlinear equations and providing that the iteration is continued until the residual $\underline{r}\,(t^p_{n+1})$ is sufficiently small this error does not have the same impact as introducing splitting at the p.d.e. level. It is however possible to obtain a rough idea of the splitting error on each iteration. Define the splitting error on the pth iteration by

$$\underline{\Delta E} = \underline{\Delta V} - \ \underline{\Delta V^*} \ . \tag{14}$$

From equations (12) and (13) this error then satisfies the equation

$$\underline{\Delta E} = \ [I - k\theta J_s]^{-1} \ h \ \theta J_f \ \underline{\Delta V} \ . \tag{15}$$

In order for the operator-splitting iteration defined by equation (13) to converge with a rate of convergence r_c it is necessary, [12] p.311, that

$$|| \ [I - k\theta J_s]^{-1} \ h \ \theta J_f \ || \quad < \ r_c$$

where $r_c < 1$. Hence, provided that the iteration is converging, a plausible estimate of the splitting error on the nonlinear equation comes from

$$||\underline{\Delta E}|| \leq \ r_c \ \ ||\underline{\Delta V}|| \ . \tag{16}$$

However as $\underline{\Delta V}$ is not known we have to use $\underline{\Delta V^*}$ in its place.

4 Space-Time Error Control.

A standard method for choosing the timestep in the numerical solution of p.d.e.s is to use a CFL stability condition. Although such a condition may ensure stability it may be imprecise as an accuracy control, particularly when complex chemistry source terms are present in the p.d.e. problem. In contrast, suppose that the local error is denoted by $\underline{l}_{n+1}(t_{n+1})$ and that a standard local error control i.e. $|| \ \underline{l}_{n+1}(t_{n+1}) \ || \ < \ TOL$ is used. In this case it is difficult to establish a relationship between the accuracy tolerance, TOL , and the global space and time errors. It is thus important to use an error control which reflects the spatial and temporal contributions to the error incurred when using the method of lines.

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,

$$\begin{aligned} \underline{E}(t) = \underline{u}(t) - \underline{V}(t) &= (\underline{u}(t) - \underline{U}(t)) + (\underline{U}(t) - \underline{V}(t)) \\ &= \underline{e}(t) + \underline{g}(t). \end{aligned} \tag{17}$$

Efficient time integration requires that the spatial and temporal are roughly the same order of magnitude. The need for spatial error estimates to be unpolluted by temporal error requires the spatial error to be the larger of the two errors. One approach for achieving this is described by Berzins [1,2] who balances the spatial and temporal errors by controlling the local time error to be a fraction of the local growth in the spatial discretization error. 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, $\underline{e}(t_n)$, is zero. A local error balancing approach is then:

$$|| \ \underline{l}_{n+1}(t_{n+1}) \ || \ < \ \epsilon \ || \ \underline{\hat{e}}(t_{n+1}) \ ||, \quad 0 < \epsilon < 1. \tag{18}$$

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

$$\underline{\dot{v}}_{n+1} \ (t) = \underline{G}_N(t, \underline{v}_{n+1}(t)), \tag{19}$$

where $\underline{v}_{n+1}(t_n) = \underline{V}(t_n)$, $\underline{\dot{v}}_{n+1}(t_n) = \underline{G}_N(t, \underline{V}(t_n))$ and where $\underline{G}_N(.,.)$ is obtained simply by setting the limiter function $\Phi(.)$ in the spatial discretization method, see equation (8), to zero and by using the first order space derivative approximations. The local-in-time space error is then estimated by

$$\underline{\hat{e}}(t_{n+1}) = \underline{V}(t_{n+1}) - \underline{v}_{n+1}(t_{n+1}) \tag{20}$$

and is computed by applying the Theta method, with one functional iteration (or one Newton-Krylov or operator-splitting iteration), to equation (19). In the case of functional iteration, equations (8) and (20) combined with the conditions on $\underline{v}_{n+1}(t_n)$ then give, [2],

$$\begin{aligned}\underline{\hat{e}}(t_{n+1}) = \theta\, k\, [\underline{F}_N(t_{n+1}, \underline{V}(t_{n+1})) - \underline{G}_N(t_{n+1}, \underline{V}(t_{n+1}))] \;+ \\ (1-\theta)\, k\, [\underline{F}_N(t_n, \underline{V}(t_n)) - \underline{G}_N(t_n, \underline{V}(t_n))].\end{aligned} \tag{21}$$

This estimate thus requires only one evaluation of $\underline{G}_N(.,.)$ per timestep. In the other cases when Newton-Krylov or operator splitting methods are used the method is very similar to the error balancing approach of Lawson, Berzins and Dew [10] for parabolic equations.

One effect of using this simple error estimate is that the effect of the source term is not taken into account as the same solution value is applied to both the o.d.e. systems defined by equations (9) and (13). Hence the source term calculation is the same in both cases. One remedy for this is to use a fixed error tolerance as the lower bound tolerance for the p.d.e. with the strong source term and to apply error balancing normally for the remaining p.d.e.s.

5 Evaluation of Error Estimate.

5.1 A Computational Example.

A number of experiments with problems in one and two space dimensions using the error control strategy given by equation (21) above are given by Berzins [1,2]. A good illustration of the performance of the error estimator is given by the standard Euler Equations Shock-Tube test problem of Sod, which is often used in the comparison of algorithms for hyperbolic equations. The problem is described fully by [6,15]. In the experiments here we consider the density ρ at time 0.2. The code of Pennington and Berzins [15] was used with the same regular mesh of 141 points as in that paper. A number of runs were performed with standard local error control with a fixed absolute tolerance of 0.5×10^{-5} and relative tolerances, Rtol, of 10^{-2}, 10^{-3} and 10^{-4}. Table 1 shows the integration statistics for the three runs. *Error* is the true error, *Espace* is the local-in-time space error estimate defined by equation (21) while *LocTErr*

Table 1
Sod Problem L1 Error Norms

Rtol.	Error	Espace	Loc T Err	NSTEPS
10^{-2}	0.55e-2	0.24e-3	0.27e-3	125
10^{-3}	0.55e-2	0.24e-3	0.18e-3	128
10^{-4}	0.53e-2	0.28e-4	0.32e-5	1028

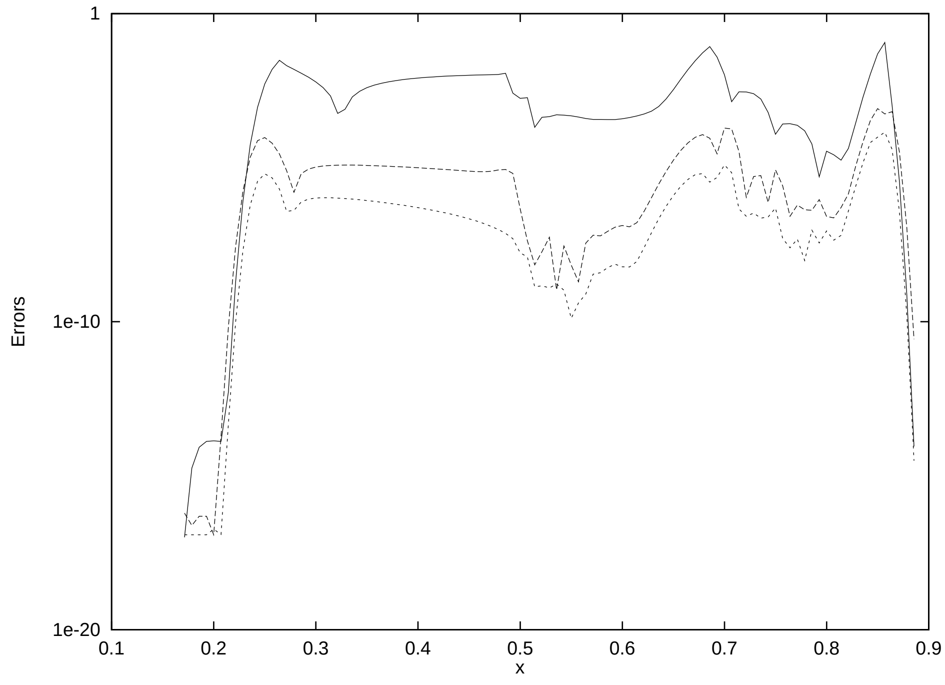

Fig. 2. Density Error Estimates for Rtol = 10^{-4} .

is the local time error estimate for the theta method, see [4]. The table shows that a large time error pollutes the spatial error estimate - by a factor of 10 in this case. This situation is illustrated by Figure 2 which shows the true error by a solid line and the local-in-time space error and the local time error by the two clearly separated dashed lines when Rtol in Table 1 is 10^{-4}.

Figure 3 shows what happens when Rtol in Table 1 is 10^{-2}. In this case both errors are larger due to an increased time error. A comparison between the exact and estimated errors shows that the error estimates do mimic the behaviour of the true error remarkably well.

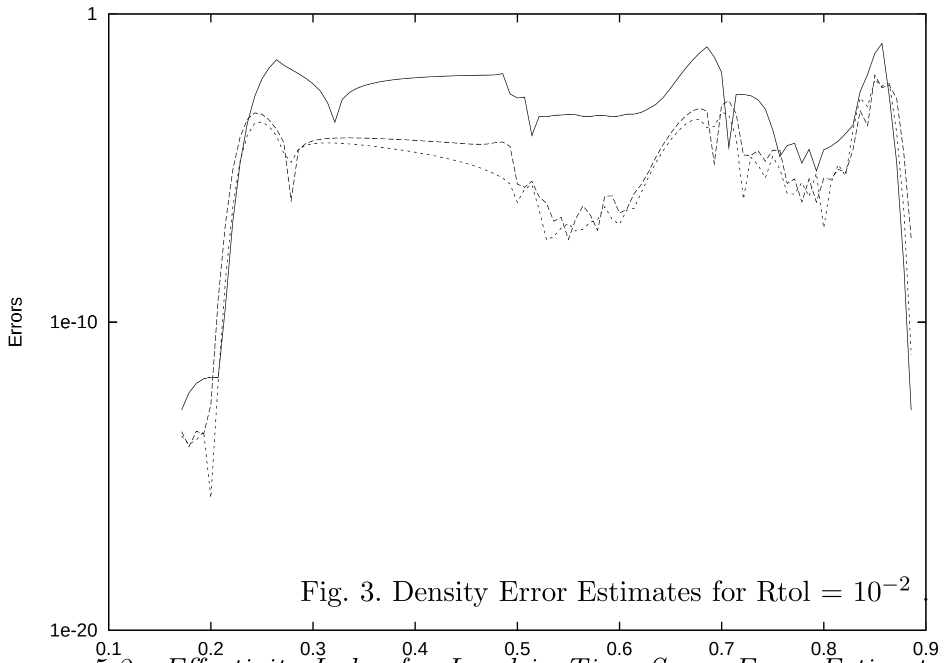

Fig. 3. Density Error Estimates for Rtol $= 10^{-2}$.

5.2 Effectivity Index for Local-in-Time Space Error Estimate.

A more formal way of assessing the performance of the error estimator defined by equation (15) is to compute the effectivity index defined by

$$e_i(t) \;=\; \frac{||\underline{\hat{e}}(t)||_1}{||\underline{\hat{e}}_{true}(t)||_1} \tag{22}$$

where $\underline{\hat{e}}_{true}(t)$ is the true local-in-time spatial error. The estimation of this is complicated in that it must reflect the true growth in the spatial error assuming that there is no error at the start of the step. In the case of p.d.e.s with known solutions it is possible to start from exact values at any time. The growth in the error over the first step is then the same as the local-in-time spatial error. For simplicity assume that the forward Euler method with a small timestep is used to compute the evolution of the error. In this case

$$\underline{\hat{e}}_{true}(t+k) \;=\; k\;[\underline{\dot{u}} \;-\; \underline{F}_N\;\;(\;t,\;\underline{u}(t)\;)] \tag{23}$$

whereas

$$\underline{\hat{e}}(t+k) \;=\; k\;[\underline{G}_N(t,\underline{u}(t)) \;-\; \underline{F}_N\;(\;t,\;\underline{u}(t)\;)] \tag{24}$$

and where $\underline{\dot{u}}$ and $\underline{u}(t)$ are vectors of exact p.d.e. solutions and their time derivatives evaluated at the spatial mesh points. From equations (23) and (24) the effectivity index is then approximated by

$$e_i(t+k) \;\approx\; \frac{||\underline{G}_N(t,\underline{u}(t)) \;-\; \underline{F}_N\;(\;t,\;\underline{u}(t)\;)||_1}{||\underline{\dot{u}} \;-\; \underline{F}_N\;(\;t,\;\underline{u}(t))||_1} \tag{25}$$

where the vector norm used is an approximation to the L_1 spatial norm. This effectivitity index thus may also be interpreted as measuring the ratio of the exact and the approximate spatial truncation errors.

5.3 Testing Procedure.

The testing procedure employed was that the error index was evaluated ten times on each of 3 square even meshes with 9x9, 27x27, 81x81 and 243x243 points using the discretization method in Berzins [2]. The meshes were constructed so that the mesh points were the centres of square cells and so that the meshes were nested. The time step used was one tenth of the mesh spacing. Two test problems were used:

1. Burgers' Equation.

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

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

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

$$w(x,t) = \frac{0.1A + 0.5B + C}{A+B+C} \quad \text{where A} = \mathrm{e}^{-0.05(\mathrm{x}-0.5+4.95\mathrm{t})/\nu},$$

$$\text{and B} = \mathrm{e}^{-0.25(\mathrm{x}-0.5+0.75\mathrm{t})/\nu)} \quad \mathrm{C} = \mathrm{e}^{-0.5(\mathrm{x}-0.375)/\nu}.$$

2. Anisotropic Test Problem.

$$\frac{\partial u}{\partial t} + \alpha\frac{\partial u}{\partial x} + (3/4+\alpha)\frac{\partial u}{\partial y} = 0, \qquad (x,y,t) \in [0,1]\times[0,1]\times(0,1.25]$$

$$u(x,y,t) = 0.25(3 + \frac{1}{1+e^{(B)}}), B = (y-x-0.75t)/8p, \quad p = 0.0001.$$

Table 2
Effectivity Index for Problems 1 and 2.

Problem	α	9x9	27x27	81x81	243x243
1		0.61	0.77	0.88	0.95
2	-0.375	1.00	1.00	1.00	1.00
2	0.750	1.00	0.99	0.98	0.88

This problem is a modified form of the one used by Zegeling [21] in his work on the Moving Finite Element Method. This problem has the interesting feature that for $\alpha > 0$ the solution wave moves at right angles to the characteristics. This means that the problem may prove difficult for codes based on approximate Riemann solvers.

The calculated values of the error index, averaged over the number of time steps, for these two problems are shown in Table 2. Similarly good results have been obtained for all the other test problems used by Berzins [2]. Despite its relatively simple nature the error estimator appears, from these early results, to work well.

6 Combustion Knock-Modelling Problem.

6.1 Problem Definition.

A very challenging problem for the method of lines is given by a simple combustion model relating to the modelling of knock in car engines. The model is used to investigate the effects of autoignition in end gasses in an idealised car engine cylinder. The onset of 'knock' is seen when large pressure pulses interact with the edges of the cylinder. A complete description of the model is provided by [13,9]. Mathematically the problem is specified by a system of five p.d.e.s representing conservation of mass, momentum and energy together with a species equation:

$$\frac{\partial \underline{u}}{\partial t} + \frac{\partial \underline{f}}{\partial x} + \frac{\partial \underline{g}}{\partial y} = \underline{h} \qquad (26)$$

where

$$\underline{u} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \\ \rho z \end{pmatrix}, \underline{f} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u[E+p] \\ \rho uz \end{pmatrix}, \underline{g} = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v[E+p] \\ \rho vz \end{pmatrix}, \underline{h} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -\rho z\ k(T) \end{pmatrix}$$

and $k(T) \;=\; exp(\beta\left(1 - \frac{1}{T}\right))$, $T = P/\rho$ and $\beta = 20.0$.

The variables ρ, u, v, p are the density, the velocities in the x and y dimensions and the pressure respectively. The variable z represents the scaled fuel concentration. The energy E is defined by the equation of state

$$E = \frac{p}{(\gamma - 1)} + \frac{\rho u^2 + \rho v^2}{2} + \alpha\rho z \quad \text{where } \gamma = 1.2 \quad \text{and } \alpha = 8.0. \tag{27}$$

The geometry of the problem and the initial condition is shown in Figure 4.

The irregular solid line in Figure 4 represents the initial position of the flame front, as taken from experimental data. The area to the left of this front contains unburnt fuel while that to the right is one in which the fuel has burnt. The dotted concentric circles indicate temperature hot spots which will lead to autoignition and pressure pulses travelling across the cylinder to cause 'knock'. Points numbered 1 to 4 are the four pressure transducers at which experimental time histories of pressure are available.

The initial conditions are as follows. The initial velocites u and v are zero. The pressure has the value $p = 1$, The fuel concentration z is zero in the burnt region and one in the unburnt region. The scaled temperature t is 0.75 in the unburnt region except at the hot spots where it rises to one and in the unburnt region it has value $1 + \alpha(\gamma - 1)/\gamma$, $\rho = p/T$. The quantities E, γ and α are defined by equation 27.

The boundary conditions are imposed through the approximate Riemann solver. This requires 'exterior' values to be defined given the interior numerical values of the variables on the boundary. These values are then used in the Riemann solver to define the fluxes at the boundary. The reflective boundary conditions are imposed by setting the exterior 'normal' velocity to be the opposite sign to the normal velocity at the boundary from the interior. The values of all the other variables on the 'exterior' being the same as the interior values. All other 'outside' solution values are the same as the interior values.

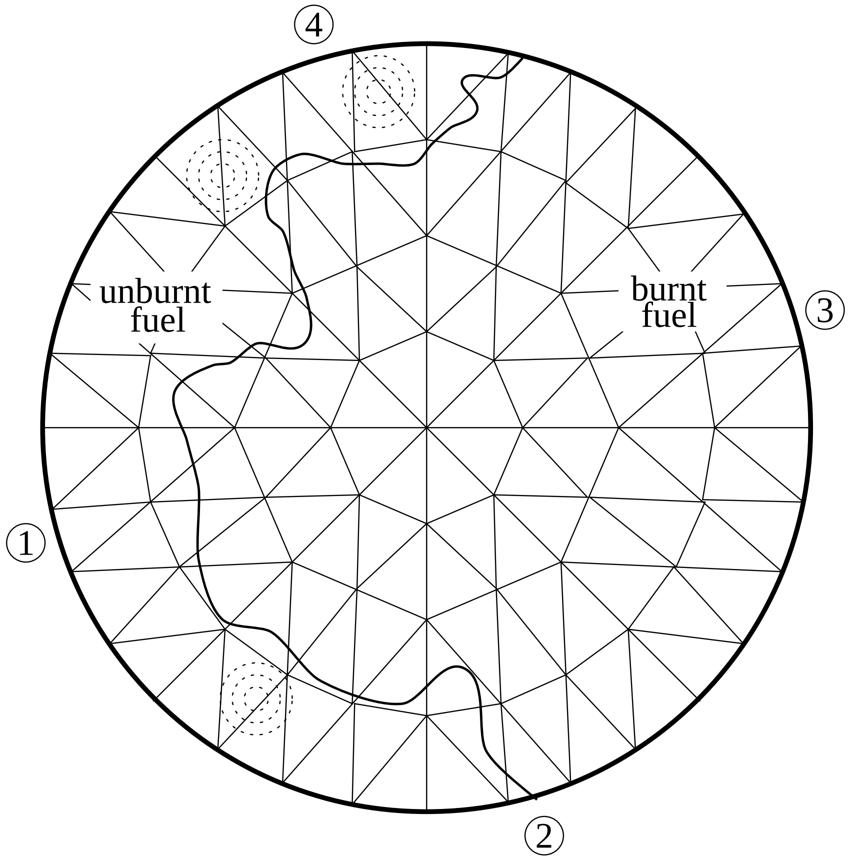

Fig. 4. Diagram of Knock Model.

6.2 Computational Experiments

The code presently used to solve this problem, LUMAD, [9,15] is based on a square regular mesh with 100x100 mesh points. This mesh makes it difficult to implement the circular boundary conditions in an accurate way. LUMAD uses the CCCT method of Gaskell and Lau [8] which may be interpreted as a flux limiter scheme, see [2]. This limiter is applied component-wise to the physical variables and an ad-hoc Riemann solver approach applied to determine the flux values. Timestepping is done using the forward Euler method with only a CFL condition to control the timestep. The entries marked TIME show the time of the peak pressure pulse at pressure transducer 1. PEAK indicates the values of this peak. When LUMAD uses 400 and 40,000 timesteps this gives CFL numbers of 1 and 0.01 respectively. The physical significance of the PEAK value is that it indicates the strength of the pressure pulse that causes 'knock' while the TIME value indicates when this occurs. Correct computation of these values is thus important if the mathematical model is to be validated against experimental observations. The LUMAD results show that convergence to the correct position and value of the pressure peak only occurs for small CFL numbers. This is the case because the CFL number only reflects the allowable

Table 3
Transducer 1 Pressure Spike

Code	*Mesh*	MODE	TSTEPS	TIME	PEAK
LUMAD	10000	CFL	400	45.23	8.61
LUMAD	10000	CFL	800	33.89	7.46
LUMAD	10000	CFL	4000	32.47	6.11
LUMAD	10000	CFL	40000	32.08	5.87
SPRINT2D	2500	LOCAL	7447	41.09	4.44
SPRINT2D	10000	LOCAL	9313	28.59	5.61
SPRINT2D	2500	BALANCE	14434	41.19	4.43
SPRINT2D	10000	BALANCE	20662	28.62	5.62

timestep for the flow to be stable and does not include the reaction terms.

The SPRINT2D code uses triangular meshes with 2500 and 10000 elements respectively. Although the code can use adaptive meshing in order to provide a fair comparison with LUMAD this option was not used. The approximate Riemann solver used by SPRINT2D was that of Suresh and Liou [20]. The SPRINT2D code was used in two modes: standard local error control and the error balancing approach described above. These modes are referred to in the results table as LOCAL and BALANCE respectively. In the BALANCE cases ϵ in equation (18) was set to 0.2 and the tolerance for the species z equation is 10^{-5}. In the LOCAL cases the local error control for SPRINT2D uses absolute tolerances of 10^{-4} for all the p.d.e. variables except the species concentration z for which 10^{-5} is used. A maximum stepsize of 5.0×10^{-4} was imposed during the initial combustion phase in order to prevent unphysical situations arising in the Riemann solver.

The initial runs with SPRINT2D switching between functional iteration and a Krylov method were disappointing in term of c.p.u. time. This is largely due to the overhead of solving the large systems of linear equations using the iterative solver. As the mesh sizes increase the situation becomes worse due to the larger systems of equations that must be solved. The situation is made worse still because of the overhead of using an unstructured mesh code and that smaller time steps are used at the start of the problem when combustion takes place. As a result of these long computation times the much faster method of solving the nonlinear equations using operator splitting described in Section 3 was developed. The recorded results for SPRINT2D in Table 3 were obtained using the new operator splitting method described below and in this case the code on the fine mesh is approximately four times as slow as LUMAD with a CFL number of 0.1 but because of the local error control uses a significantly smaller timestep while combustion takes place and a larger timestep when combustion

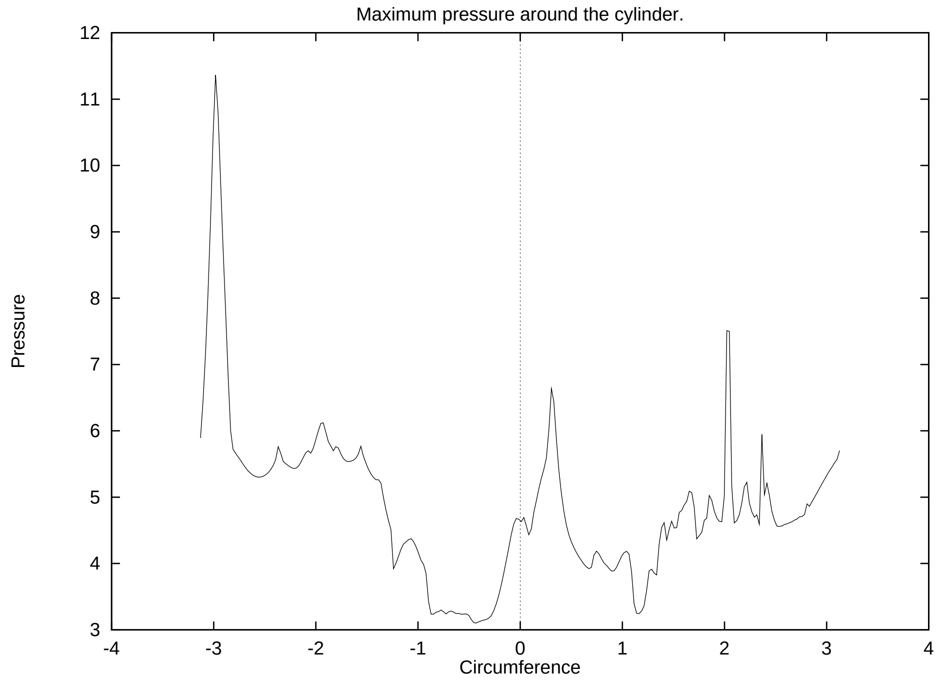

Fig. 5. Maximum boundary pressures.

has finished.

In the case when error balancing is used yet more timesteps are used and the code is slower yet again by a factor of about two. The difference now is that in the runs with ordinary local error control once combustion takes place the timestep seems to be determined by the need for functional iteration to converge. In contrast when error balancing is used then the spatial error growth is used to compute a tighter tolerance for the flow part of the problem. In both cases when the SPRINT2D code is applied with a coarse mesh the time at which the peak arrives is much later and the peak value is too small, though the time of the peak value is not as different from that obtained by LUMAD with a CFL number of one. The table shows that unless great care is taken with the choice of time step and spatial mesh over-large pressure pulses at incorrect times may be recorded. In particular LUMAD appears to need a very small CFL number (< 0.01) to produce similar results to SPRINT2D.

The important aspect of the numerical results is the size of any pressure spikes to reach the boundary. The maximum recorded values of the pressure on the boundary obtained by using SPRINT2D on the fine mesh ofg 10,000 elements are shown in Figure 5. Zero radians corresponds to 'east' on Figure 4 with positive radians going anti-clockwise to 'west' and negative radians going clockwise to 'west'. Of great interest is the largest pressure spike caused by the interaction of pressure waves coming from the bottom two exothermic centers in Figure 4 and joining at the boundary.

7 Conclusions.

In this paper we have described the components of a method of lines solver for convection problems and have extended the solver to a complex combustion problem. A traditional method of lines approach has been modified to include spatial and temporal error balancing and operator splitting. Early results suggest tha the method is more reliable than fixed space and time step methods and provided that the splitting approach is used does not have as high a computational cost as a traditional m.o.l. code. Future work will concentrate on applying the adaptive space mesh methods we have employed elsewhere to the combustion problem.

Acknowledgement. The authors would like to thank Shell Research Ltd. and the EPSRC for funding and their many collaborators on the Knock Modelling Project for their help, notably J.Pan and C.G.W. Sheppard.

References

[1] M. Berzins, Balancing Space and Time Errors for Hyperbolic Equations, in L.S. Xanthis, ed., *Method of Lines and Dimensional Reduction in Computational Mathematics and Mechanics* (Kluwer Academic Pub. to appear 1995).

[2] M. Berzins, Temporal error control for convection-dominated equations in two space dimensions. *SIAM J. of Sci. Comput.* **16** (1995) 558–581.

[3] M. Berzins and R.M. Furzeland. An adaptive theta methods for the solution of stiff and non-stiff differential equations. *Appl. Numer. Math.* **9** (1992), 1–19.

[4] M. Berzins and J.M. Ware, Reliable Finite Volume Methods for the Navier Stokes Equations, in F-K Hebeker, R. Rannacher and G.Wittum, eds., *Numerical Methods for the Navier Stokes Equations,* (Notes on Numerical Fluid Mechanics **47**, Vieweg.) 1–8.

[5] M. Berzins and J.M. Ware, Positive discretization methods for hyperbolic equations on irregular meshes. *Appl. Numer. Math.* **16** (1995), 417–438.

[6] R. Biswas, J.E. Flaherty and D.C. Arney. An Adaptive Mesh Moving and refinement procedure for one dimensional conservation laws. *Appl. Numer. Math.* **11** (1993) 259–282.

[7] T.J. Chung (ed). Numerical Modeling in Combustion. Taylor and Francis, Washington D.C. (1993).

[8] P.H. Gaskell and A.K.C. Lau Curvature compensated convective transport: SMART - a new boundedness preserving scheme. *Internat. J. Numer. Methods Fluids.* **8** (1988) 617–641.

[9] G. Konig, R.R. Maly, D. Bradley, A.K.C. Lau and C.G.W. Sheppard Role of Exothermic Centres on Knock Initiation and Knock Damage. *S.A.E. paper* 90-2136 (1990). S.A.E. International, Warrendale, PA 15096-0001, USA

[10] J. Lawson, M. Berzins and P.M. Dew , Balancing Space and Time Errors for Parabolic Equations, *SIAM J. Sci. Statist. Comput.* **12** (1991) 573–594.

[11] P.K. Moore and J.E. Flaherty, Adaptive overlapping grid methods for parabolic systems in two space dimensions, *J. of Comput. Phys.* **98** (1992) 54–63.

[12] J.M. Ortega and W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, London and New York, 1970.

[13] J. Pan and C.G.W. Sheppard, A theoretical and experimental study of the modes of end gas autoignition leading to knock in an SI engine. *S.A.E. paper* 94-2060 (1994). S.A.E. International, Warrendale, PA 15096-0001, USA

[14] S. Osher and F. Solomon, Upwind difference schemes for hyperbolic systems of conservation laws. *Math. Comp.* **38** (1982) 339–374.

[15] S.V. Pennington and M. Berzins, New NAG Library Software for First-Order Systems of Time-Dependent P.D.E.s. *ACM Trans. on Math. Soft.* **20** (1994) 63–99.

[16] P.L. Roe, Approximate Riemann Solvers, parameter vectors and difference schemes. *J. of Comput. Phys.* **43** (1981) 357–372.

[17] W.L. Seward and K.R. Jackson, Adaptive linear equations solvers in codes for large stiff systems of o.d.e.s . *SIAM J. Sci. Comp.* **14** (1993) 800–823.

[18] T. Strouboulis and J.T. Oden, A Posteriori Error Estimation of Finite Element Approaches in Fluid Mechanics. *Comp. Meths. Appl. Mech. and Eng.* **78** (1990) 201–242.

[19] 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.

[20] A. Suresh and M-S. Liou, The Osher Scheme for Non-Equilibrium Reacting Flows. *Internat. J. Numer. Methods Fluids.* **15** (1992) 219–232.

[21] 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).