Towards an Automatic Algorithm for the Numerical Solution of Parabolic Partial Differential Equations Using the Method of Lines

J. Lawson and M. Berzins

University of Leeds

1 Introduction

An algorithm which uses the Method of Lines to solve parabolic partial differential equations (PDEs) in one space dimension is presented. The spatial mesh used in the spatial discretization of the PDE is modified by the algorithm in such a way that the energy norm of the relative spatial discretization error is controlled with respect to a user specified tolerance. An accuracy tolerance to be used in the solution of the ordinary differential equations (ODEs) is calculated in order that the spatial discretization and global time errors are approximately balanced. This ensures that the Method of Lines can be used in an automatic way to solve parabolic PDEs in one space dimension. The results of numerical experiments are used to illustrate the performance of this algorithm.

The Method of Lines is widely used in general purpose software for the numerical solution of parabolic PDEs. In the Method of Lines the spatial derivatives in the PDE are approximated at each point of a spatial mesh using, for example, finite difference or finite element techniques. This results in a system of time dependent ODEs which can be solved using an ODE solver. The accuracy of the Method of Lines is influenced by the spatial discretization method used, the spatial mesh and the method of integration of the ODEs. The spatial discretization points should be chosen so that the computed solution accurately models the true solution. The ODEs should be integrated with just sufficient accuracy so that the global time error does not significantly corrupt the spatial accuracy. In existing Method of Lines software, the user usually supplies a spatial mesh at the start of the computation. More recent packages, such as SPRINT [4], may then modify this spatial mesh during the computation. Most adap-

1992

tive techniques currently available do not, however, attempt to control the spatial discretization error explicitly. In the time integration, the standard procedure is to control the local time error per step (LEPS) with respect to an accuracy tolerance, *tol*. In general, this tolerance cannot be easily related to the spatial discretization error.

The purpose of this paper is to present an automatic algorithm which uses the Method of Lines to solve parabolic PDEs in one space dimension. The algorithm requires the user to provide only a specification of the problem, an initial spatial mesh and a spatial discretization error tolerance. In the algorithm presented here, it is the energy norm of the relative spatial discretization error that is to be controlled with respect to a user supplied tolerance, *EPS*. This control is achieved through the use of an $h - refinement$ adaptive spatial mesh method, based on that proposed by Bieterman and Babuska [5]. The novel feature of the algorithm is that the accuracy tolerance used in the integration of the ODEs is calculated automatically. It is calculated in such a way that the spatial discretization and time integration errors are of the same order of magnitude, but so that the spatial discretization error dominates the time integration error. Other approaches in a similar spirit are those of Babuska and Luskin [1], Johnson [6] and more particularly Schoñauer, Schnepf and Raith [9]. The relative merits of these approaches are discussed by Lawson [8].

The paper is structured in the following way. Section 1 describes the Method of Lines and indicates how the global error may be decomposed and estimated, Berzins [3]. This allows the error control strategy for the time integration, proposed by Lawson, Berzins and Dew [7], to be summarized in Section 2. Section 3 contains a summary of the adaptive spatial mesh algorithm used to control the spatial error, while Section 4 explains how this algorithm is modified to work with the error balancing approach. Finally, in Section 5, the results of numerical experiments are presented which show that the algorithm successfully controls the spatial discretization error, and as a result, also the time integration error, automatically. This algorithm appears to be a promising start to developing fully automatic mathematical software for the numerical solution of PDEs.

1.1 Problem Class and Method of Solution

The class of parabolic PDEs to be considered is written as

$$\begin{aligned} c(x,t)\frac{\partial u}{\partial t} &= \frac{\partial}{\partial x}\, r\left(x,t,u,\frac{\partial u}{\partial x}\right) + f\left(x,t,u,\frac{\partial u}{\partial x}\right), \\ (x,t) \in \Omega &= [a,b] \times (0,t_e] \end{aligned} \tag{1.1}$$

and it is assumed that there exist real constants c_1 and c_2 such that

$$0 < c_1 < c(x,t) < c_2 \text{ for all } (x,t) \in \Omega \tag{1.2}$$

For notational convenience and the sake of brevity, it is assumed that only one PDE is to be solved and that the boundary conditions are of the form

$$g_a(t, u(a,t)) = 0, \qquad g_b(t, u(b,t)) = 0 \tag{1.3}$$

for $t \in (0, t_e]$. The initial condition has the form

$$u(x,0) = u_0(x), \qquad x \in [a,b]. \tag{1.4}$$

The extensions of the methods discussed in this paper to systems of PDEs or to problems with Neumann or mixed boundary conditions are straightforward, [8]. It is assumed that the PDE defined by the above equation is well-posed and has a unique continuous solution $u(x,t)$, for all $(x,t) \in \Omega$.

The PDE is to be approximated at each point of the spatial mesh

$$\delta : a = x_1 < x_2 < \cdots < x_N = b, \tag{1.5}$$

which has $N-1$ sub intervals of length h_j, where $h_j = x_{j+1} - x_j, \qquad j = 1, 2, \ldots, N-1$. The spatial discretization scheme used here, to discretize the PDE in space, is a modified version of the box scheme, [11], which can also be thought of as a Galerkin Finite Element method with quadrature. The method can be written as

$$\begin{aligned}(h_{j-1}c_{j-1/2} + h_j c_{j+1/2})\frac{\partial U}{\partial t} &= 2(R_{j+1/2} - R_{j-1/2}) \\ &+ (h_{j-1}f_{j-1/2} + h_j f_{j+1/2})\end{aligned} \tag{1.6}$$

where $R_{j+1/2}$ and $R_{j-1/2}$ are defined by

$$\begin{aligned}R_{j+1/2} &= r(\frac{x_j + x_{j+1}}{2}, t, \frac{U(x_j,t) + U(x_{j+1},t)}{2}, \\ &\quad \frac{U(x_{j+1},t) - U(x_j,t)}{h_j}), \qquad j = 2, \ldots, N-1 \\ R_{j-1/2} &= r(\frac{x_j + x_{j-1}}{2}, t, \frac{U(x_j,t) + U(x_{j-1},t)}{2}, \\ &\quad \frac{U(x_j,t) - U(x_{j-1},t)}{h_{j-1}}), \qquad j = 2, \ldots, N-1\end{aligned}$$

The quantities $c_{j+1/2}$, $c_{j-1/2}$, $f_{j+1/2}$ and $f_{j-1/2}$ are defined similarly and $U(x_j,t)$ is the approximate solution defined by the spatial discretization method at the point x_j. The boundary conditions at the points $x = a$ and $x = b$ reduce to the algebraic equations

$$g_a(t, U(x_1,t)) = 0, \qquad g_b(t, U(x_N,t)) = 0 \tag{1.7}$$

and the initial condition is defined by evaluating the function $u_0(x)$ at the spatial mesh points

$$U(x_j, 0) = u_0(x_j), \qquad j = 1, \ldots, N.$$

This spatial discretization scheme results in a system of differential algebraic equations, given by equations (1.6) and (1.7), which can be written as the initial value problem(i.v.p.)

$$A_N(t)\dot{\mathbf{U}} = \mathbf{F}_N(t, \mathbf{U}(t)), \tag{1.8}$$

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

$$\mathbf{U}(t) = [U(x_1, t), U(x_2, t), \ldots, U(x_N, t)]^T,$$

and $A_N(t)$ is an $N \times N$ matrix defined in [3]. In practice equation (1.8) is a stiff or mildly stiff system of time dependent ODEs that can be solved using software based on the backward differentiation formulae (b.d.f.), e.g. SPRINT [4], to compute the approximation, $\mathbf{V}(t)$, to the true solution, $\mathbf{u}(t)$, of the PDE. The global error in the numerical solution can be expressed as the sum of the spatial discretization error, $\mathbf{es}(t) = \mathbf{u}(t) - \mathbf{U}(t)$, and the global time error, $\mathbf{ge}(t, TOL) = \mathbf{U}(t) - \mathbf{V}(t)$. That is,

$$\begin{aligned} E(t) &= \mathbf{u}(t) - \mathbf{V}(t) = (\mathbf{u}(t) - \mathbf{U}(t)) + (\mathbf{U}(t) - \mathbf{V}(t)) \\ &= \mathbf{es}(t) + \mathbf{ge}(t, TOL). \end{aligned} \tag{1.9}$$

This can be estimated using the method of Berzins [3], which approximates the global error at t_{n+1} by

$$\mathbf{E}(t_{n+1}) = M^{-1}(A_N(t_{n+1})\mathbf{E}(t_n) + k_n\mathbf{TE}_{n+1}) + \mathbf{le}_{n+1}(t_{n+1}, TOL), \tag{1.10}$$

where $\mathbf{E}(t_n)$ is the global error incurred at the start of the current time step, $k_n = t_{n+1} - t_n$. The spatial truncation error at t_{n+1} is denoted by $\mathbf{TE}_{n+1}$ (the estimation of which requires the spatial discretization error to dominate the time integration error). The Jacobian matrix $M = A_N - k_n\gamma J$, $\left(J = \frac{\partial \mathbf{F}_N}{\partial \mathbf{U}}\right)$, is computed by the ODE solver and stored in its LU decomposed form. Finally, $\mathbf{le}_{n+1}(t_{n+1}, TOL)$ is the local time error per step.

1.2 Error Control in Codes for Solving Stiff ODEs

Most codes for solving time dependent ODEs control either the local time error per step, (LEPS), with respect to a user supplied accuracy tolerance, TOL, or the local time error per unit step (LEPUS), $\frac{\mathbf{le}_{n+1}(t_{n+1}, TOL)}{k_n}$. When controlling the LEPS it is difficult to establish a relationship between the accuracy tolerance, TOL, and the global time error, Shampine [10]. On

the other hand, if the LEPUS is controlled then it can be shown, Stetter [12], that the computed solution satisfies tolerance proportionality, that is

$$\mathbf{ge}(t, TOL) = \mathbf{v}(t)TOL + o(TOL),$$

where $\mathbf{v}(t)$ is independent of TOL and $\mathbf{v}(t)$ and $\mathbf{v}'(t)$ are bounded on $[0, t_e]$. Here, $o(TOL)$ is the notation used in [12] to denote a term that is numerically negligible compared with terms of order TOL in the same equation. Although LEPUS control is generally thought to be inefficient for standard stiff ODEs, there is a fundamentally different situation in the Method of Lines in that the time error control strategy must take account of the spatial discretization error already present.

2 Balancing the Space and Time Errors in the Method of Lines

In order that the Method of Lines be used efficiently, the time integration error should not dominate the error due to the spatial discretization of the PDE nor should the ODEs be integrated with a much higher degree of accuracy than that already attained in space. It follows that the spatial and temporal errors should be balanced, although in practice the spatial discretization error must actually dominate so that the estimate of the spatial truncation error remains accurate, [3]. However, it is difficult to select a priori an ODE tolerance that will ensure that this is so. This is particularly difficult if the LEPS is controlled by the integrator, since the global time error need not be related to the chosen accuracy tolerance. In addition, the spatial accuracy may vary with time, so any fixed ODE tolerance is unlikely to be related to the size of the changing spatial discretization error. This suggests that an error control strategy in which an accuracy tolerance, related to the spatial discretization error, is automatically selected by the algorithm is required for the integration of the ODEs. In addition, this tolerance must be adjusted as the spatial discretization error varies. It is the global time error that is to be balanced with the spatial discretization error which suggests that some form of LEPUS control must be used. One such strategy has been developed by Lawson, Berzins and Dew [7] by considering the relative contributions of the local time error and the spatial discretization error to the global error in the numerical solution.

The strategy of [7] uses the approximation of the global error, equation (1.10), to control the local time error to be a fraction of the growth in the spatial discretization and existing errors over the interval $[t_n, t_{n+1}]$, that is,

$$\|\mathbf{le}_{n+1}(t_{n+1}, TOL)\| < \varepsilon \|\mathbf{E}(t_{n+1}) - \mathbf{E}(t_n) - \mathbf{le}_{n+1}(t_{n+1}, TOL)\|. \quad (2.1)$$

It can be shown, [7], that, for a suitable value of ε, this yields a time integration error which is dominated by the spatial discretization error.

That this is a form of LEPUS control can be seen more easily by considering the implementation of the strategy (2.1) in which equation (1.10) is used to substitute for $\mathbf{E}(t_{n+1})$. Thus, the local time error can be controlled according to

$$\|\mathbf{le}_{n+1}(t_{n+1}, TOL)\| < \varepsilon \|M^{-1}(\mathbf{E}(t_n) + k_n \mathbf{TE}_{n+1}) - \mathbf{E}(t_n)\| \qquad (2.2)$$

which, on using the definition of the matrix M, gives

$$\|\mathbf{le}_{n+1}(t_{n+1}, TOL)\| < k_n \varepsilon \|M^{-1}(\gamma J \mathbf{E}(t_n) + \mathbf{TE}_{n+1})\| \qquad (2.3)$$

This expression shows how the accuracy tolerance used in the LEPUS control strategy varies with the spatial truncation error and the global error. A more complete description of the control strategy is given in [7].

3 Control of the Spatial Discretization Error

The adaptive mesh method adopted is based on the adaptive finite element Method of Lines (FEMOL) algorithm proposed by Bieterman and Babuska, [5]. This section provides an outline of the strategy, further details about the adaptive FEMOL are to be found in [5] and [8]. The aim of the strategy is to control the energy norm of the relative spatial discretization error with respect to a user specified error tolerance, EPS. That is, the spatial mesh is modified in order that the vector of the spatial discretization error, $\mathbf{es}(t)$, when mapped onto a piecewise linear function in space, $\mathbf{es}(t, x)$, satisfies

$$\frac{|||es(t,\cdot)|||}{|||u(t,\cdot)|||} \leq EPS \qquad (3.1)$$

where $|||v(t,\cdot)|||$ is the energy norm defined as $(\int_a^b v_x^2 dx)^{1/2}$ on the appropriate space, [5]. An important feature of this algorithm is that the error is *controlled* while constructing meshes that are also suitable for a number of future time steps.

3.1 Estimation of the Spatial Error

An estimate, $\epsilon(t)$, of $|||es(t,\cdot)|||$ can be given by, see Babuska and Rheinboldt [2],

$$\epsilon(t) = \left\{ \sum_{n=1}^{N-1} |\eta_n(t)|^2 \right\}^{1/2}, \qquad (3.2)$$

where $|\eta_n(t)|^2$ is the local error indicator for the spatial element $[x_n, x_{n+1}]$. This estimate can also be used with, for example, the finite element like modified box discretization scheme for non-polar parabolic equations given

by Skeel and Berzins [11]. The following piecewise constant function is used in constructing the new mesh

$$w(t,x) \equiv h_n^{-1}(12|\eta_n(t)|^2)^{1/3},\ x \in (x_n, x_{n+1}), \qquad n = 1,\ldots,N-1 \quad (3.3)$$

and is related to the second spatial derivative of the solution, [5].

When a mesh is modified, both (or either of) its shape and intensity may change. The shape of a mesh defines the general distribution of the mesh points identifying the areas where the points should be clustered and the areas where relatively few points are needed. The shape of a mesh can be described through a mesh function, defined by

$$\xi_\delta(x) \equiv \begin{cases} 1/h_n : x \in (x_n, x_{n+1}),\ n = 1,\ldots,N-1 \\ (1/2)(1/h_{n-1} + 1/h_n) : x = x_n,\ n = 1,\ldots,N-1 \end{cases}$$

The intensity of a mesh is the number of points actually in the mesh.

From the definition of $\epsilon(t)$, (3.2), it can be seen that this estimate depends on the function $w(t,\cdot)$ and the mesh function describing the present mesh, that is,

$$\epsilon(t) = \left(\frac{1}{12}\int_\Omega \frac{w^3(t,x)}{\xi_\delta^2(x)}dx\right)^{1/2}. \quad (3.4)$$

3.2 Construction of the New Mesh

Consider a time $t = t_m$ and a current mesh δ. Suppose that

$$\epsilon(t_m) > 0.95\ EPS\ |||V(t_m,\cdot)|||$$

so that a new mesh, δ^+ say, must be constructed from the current mesh. The new mesh is chosen so that the predicted error on the new mesh at the time step immediately after remeshing, that is at $t = t_m^+$, satisfies

$$\epsilon(t_m^+) \approx EPSDN\ |||V(t_m^+,\cdot)|||, \qquad EPSDN \in [0.6, 0.8]\ EPS \quad (3.5)$$

where $EPSDN$ is chosen adaptively each time remeshing occurs. The underlying assumption in this adaptive process is that the introduction of extra mesh points will cause the error to decrease. There are three stages to the formation of the new mesh. The formation of a suitable mesh function defining the shape of the new mesh, ξ_{δ_+}. This is achieved using a pattern recognition technique and heuristics to predict where the mesh points will be needed for future time steps, see [5] for details. The multiplication of this mesh function so that the mesh has the required intensity for the requirement (3.5) is to be satisfied. The addition and subtraction of points from the current mesh to yield a mesh whose shape resembles the shape of

ξ_{δ_+} and which has the required intensity for (3.5) is to be satisfied. Details of exactly how this is accomplished can be found in [5]. The computed solution is then interpolated onto the new mesh and the time integration restarted using the approach adopted in [4].

4 Towards an Automatic Algorithm

The automatic algorithm basically consists of the error control strategies described in the previous two sections. In order that they work together efficiently, modifications to each strategy are required.

The input required from the user consists only of the problem specification, an initial spatial mesh (which is usually coarse and uniform) and an error tolerance for the spatial discretization error, EPS. The PDE is discretized in space using the modified box scheme [11] and the resulting system of time dependent ODEs is integrated using the SPGEAR (b.d.f.) module of SPRINT [4]. The time integration routine must be suitably modified to use the LEPUS control strategy given by (2.1). Rather than test for possible remeshing only at user specified times, [5], at *each* time step the estimate $\epsilon(t)$ of $|||es\ (t,\cdot)|||$ is calculated, and if

$$\epsilon(t) > 0.95\ EPS\ |||V(t,\cdot)|||$$

then a new mesh is constructed that ensures that

$$\epsilon(t^+) \approx EPSDN\ |||V(t,\cdot)|||.$$

The remeshing strategy is more efficient if $EPSDN$ is allowed to vary and so it is chosen adaptively at each remeshing time. The actual value given to $EPSDN$ is governed by how much the estimate of the energy norm of the relative spatial discretization error fails to satisfy the error test (3.1). That is,

$$EPSDN = \begin{cases} 0.8\ EPS & \text{if} \quad 0.95 \leq R_\epsilon \leq 0.975 \\ 0.7\ EPS & \text{if} \quad 0.975 < R_\epsilon \leq 1 \\ 0.6\ EPS & \text{if} \quad R_\epsilon > 1 \end{cases}$$

where $R_\epsilon(t) = \frac{\epsilon(t)}{|||V(t,\cdot)|||\ EPS}$. In order that remeshing does not occur too frequently and to allow the stepsize and order of the ODE solver to increase, a heuristic constraint is included to enforce at least 10 time steps between remeshings (unless $\epsilon(t) > 1.75\ EPS\ |||V(t,\cdot)|||$, in which case remeshing is allowed). This restriction can result in a violation of the user supplied error tolerance. In practice this has been found to be preferable to the possibility of remeshing at every time step, in which case the influence of interpolation errors becomes rather large. The estimation of the spatial truncation error, **TE**, used in the global error estimator of Berzins [3],

requires pairs of adjacent spatial element sizes to be equal. To obtain such meshes, remeshing takes place on a coarse mesh defined by

$$\delta^c : a = z_1 < z_2 < \cdots < z_M = b$$

where $z_i = x_{2i-1}$, $i = 1, \ldots, M$, $M = (N+1)/2$, N is odd.

The selection of the mesh function and subsequent construction of the new mesh is carried out using the technique described by Bieterman and Babuska [5]. The mesh on which the solution is to be computed after remeshing, is found by simply bisecting each element in the newly formed coarse mesh. Once a new mesh has been found, the computed solution and the Nordsieck history array used by the time integrator are interpolated, using a complete cubic spline interpolant, onto this mesh and the time integration is restarted. A flying restart, which uses the same stepsize and order used immediately before remeshing, is performed. This is often faster than performing a full restart, but there is an increased risk of convergence failures. This risk is minimised by keeping the intensity of consecutive meshes constant, unless a significant change (greater than ten percent) in the number of points is required. A full restart will be performed automatically by SPRINT [4] in the event of repeated convergence failures.

Since the accuracy tolerance for the time integration over the next time step depends partially on the error incurred prior to spatial remeshing, this tolerance must be modified according to the expected reduction in the spatial discretization error. Once the time integration has been restarted, the time integration proceeds until the next point where remeshing is required is reached or the end of the computation, whichever is soonest.

5 Numerical Experiments

In this section, it will be shown that it is possible for the new automatic algorithm to be competitive with other algorithms based on the Method of Lines. Although the automatic algorithm has been used to solve a number of parabolic problems, here, for reasons of brevity, only one example will be used to demonstrate the performance of the algorithm. Full details of more extensive numerical testing are given in [8]. The example used is Burgers' equation defined by

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2} - u \frac{\partial u}{\partial x}, \ (x,t) \in (0,1) \times (0,1], \ \nu = 0.005,$$

with Dirichlet boundary conditions and the initial condition consistent with the analytic solution

$$u(x,t) = \frac{0.1A + 0.5B + C}{A + B + C}$$

Table 1. Using the LEPS strategy and a fixed mesh

PTS	TOL	GERR	REST	CP	NST	FN	J	AVG
41	0.1D-3	0.12D00	0.31D00	4	99	255	9	0.51
81	0.5D-4	0.36D-1	0.69D-1	9	144	367	13	0.65
161	0.1D-4	0.91D-2	0.50D-1	20	196	485	16	0.72
321	0.5D-5	0.22D-2	0.30D-1	53	241	621	25	0.81

Table 2. Using the LEPUS strategy and a fixed mesh

PTS	GERR	REST	CP	NST	FN	J	AVG
41	0.12D00	0.31D00	4	62	197	7	0.51
81	0.38D-1	0.72D-1	8	94	260	12	0.61
161	0.96D-2	0.50D-1	19	140	397	21	0.69
321	0.24D-2	0.30D-1	47	185	506	24	0.73

PTS number of spatial mesh points.

TOL accuracy tolerance for LEPS strategy.

EPS spatial discretization error tolerance.

GERR maximum global error in the computed solution.

REST maximum estimate of relative error, $\epsilon(t)/|||V(t,\cdot)|||$.

CP CPU time in seconds on an Amdahl 5860.

NST number of ODE time steps used.

FN number of function evaluations made.

J number of Jacobian evaluations made.

AVG average value of the global error index, see below.

REM number of remeshings.

Figure 1. Key to Tables 1, 2 and 3

Table 3. Using the LEPUS strategy and remeshing

MIN/MAX PTS	EPS	GERR	REST	CP	NS	FN	J	AVG	REM
41/45	0.30	0.11D00	0.31D00	5	77	283	23	0.56	7
41/83	0.07	0.62D-1	0.67D-1	10	124	475	42	0.63	7
81/111	0.05	0.26D-1	0.50D-1	11	120	404	29	0.70	12
81/177	0.03	0.16D-1	0.34D-1	18	160	491	30	0.71	28

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

The testing procedure adopted was as follows. First, the problem was solved using fixed evenly spaced meshes, with the LEPS controlled in the ODE integration with respect to a user supplied tolerance. Table 1 shows these results. The accuracy tolerances used were the coarsest tolerances that could be found through repeated experimentation, such that the spatial discretization errors dominate the time integration errors. In this case, if the user wishes to obtain a computed solution with a certain degree of accuracy, the problem has to be repeatedly solved using different spatial meshes and with different accuracy tolerances for the ODE integration, until a suitable solution is obtained.

Second, to assess the performance of the new LEPUS control strategy, the problem was solved on fixed evenly spaced meshes, with the LEPUS controlled, according to (2.1), in the ODE integrator. The results are shown in Table 2. In this case, the user must repeat the computation using different spatial meshes until a solution with the desired accuracy is calculated. The accuracy tolerance used in the LEPUS control strategy (2.1) is computed automatically by the algorithm in such a way that the time integration error does not dominate the spatial discretization error. The parameter *eps* in the strategy (2.1) was given the conservative value 0.3; reasons for this choice can be found in [7] and [8].

Table 3 contains the results obtained when using the automatic algorithm described in Section 4 to solve the test problem. In this algorithm, the spatial mesh is adapted so that the estimate of the energy norm of the relative spatial discretization error in the computed solution satisfies the user's error tolerance EPS. The ODEs are integrated using the LEPUS control strategy (2.1) which ensures that the spatial discretization error dominates the time integration error. The values given to the error tolerance, EPS, are the values of the energy norm of the spatial discretization errors incurred when solving the problem on fixed, evenly spaced meshes (see Tables 1 and 2). Unlike the previous two algorithms, the user does not have to experiment to obtain a solution of desired accuracy, instead

the user has to simply specify the required accuracy in terms of an energy norm error tolerance. The automatic algorithm also provides the user with estimates of the global error in the computed solution. In addition, although the new algorithm cannot claim to be always as computationally efficient as other approaches, the results below show that the overheads of remeshing do not increase the cost dramatically. The approximation (3.2) estimated the error to within 10% of its true value and the performance of the global error indicator (1.10) was measured using the index

$$I = \frac{\| \textit{ Estimated global errors at time } t \|_\infty}{\| \textit{Actual global errors at time } t \|_\infty}.$$

The results from Tables 1 and 2 show that when employing either the LEPS or the new LEPUS strategy, (2.1), solutions of comparable accuracy are obtained. The efficiency of the computations can be compared by considering the number of ODE time steps used, the number of function and Jacobian evaluations and the CPU time required for each run. It can be seen that it is more efficient to use the LEPUS strategy which calculates the accuracy tolerance at each time step. Similar results can be found in [7] and [8]. However, this result is typical of the performance of this new strategy. Difficulties occur if the global error estimate, (1.10), is larger than the actual error. This may happen if the mesh is too coarse or the spatial discretization error so large that the estimate of the spatial truncation error is no longer reliable. In which case the calculated accuracy tolerance may not ensure that the spatial discretization error dominates. In such cases, it is possible to use a comparison between the error estimate given by (3.2) and the global error estimate given by (1.10) to finely tune the parameter ϵ used in the LEPUS control strategy (2.1). This is shown in [8].

The aim of the mesh modification technique used within the automatic algorithm is to control the estimate of the energy norm of the spatial discretization error with respect to the user specified tolerance, EPS. The results in Table 3 show that the required control has been achieved. Although, for $EPS = 0.3$ and 0.03, the error in the computed solution does not quite satisfy the tolerance. This is due to the heuristic restriction which enforces each mesh to remain fixed for at least ten time steps. When considering this error, the new automatic algorithm compares favourably, in terms of accuracy and efficiency, with the fixed mesh results presented in Tables 1 and 2. In addition, the user no longer has to experiment with different spatial meshes and accuracy tolerances for the ODE integration. Instead, the user can now specify the required accuracy in the computed solution and in most cases a suitable solution can be computed on the first attempt. The actual remeshing process is not computationally expensive, but the restarting process can be. This is reflected in the increased number of function and, more particularly, Jacobian evaluations performed. However, the computational cost of a function call in Table 3 does not match

Table 4. Using the LEPUS strategy and remeshing

MIN/MAX PTS	EPS	GERR	REST	CP	NST	FN	J	AVG	REM
81/261	0.0175	0.35D-2	0.19D-1	47	305	1139	102	0.61	15
81/277	0.0160	0.17D-2	0.15D-1	49	260	874	65	0.66	9
81/361	0.0110	0.91D-3	0.11D-1	77	327	1117	88	0.53	12

the computational cost of a function call in the corresponding entries in Tables 1 and 2. This is because the size of the ODE system varies in the experiments shown in Table 3.

The accuracy of the computed solution can also be assessed through the global error estimate, (1.9) which is used in the LEPUS control strategy (2.1). A comparison between the results in Tables 1, 2 and 3 shows that, when controlling the estimate of the energy norm of the relative spatial discretization error, control over the L_∞ norm of the global error cannot be guaranteed. It is possible, when using the automatic algorithm, to compute solutions with global errors comparable to those incurred when using fixed spatial meshes. This can be achieved by using different values of EPS. The results presented in Table 4, show that the automatic algorithm remains competitive, in terms of efficiency, when computing solutions with global errors similar to those incurred when using fixed spatial meshes. In addition, the energy norm of the relative spatial discretization error is smaller.

Again, further results and a more detailed discussion of these results, when using the automatic algorithm, can be found in [8].

6 Conclusions

The results obtained when applying the automatic algorithm to a limited class of PDEs indicate that the algorithm is a promising start to developing Method of Lines codes which automatically control the error in the computed solution. The adaptive mesh strategy used gives good control over the relative spatial discretization error without having to remesh too frequently. The LEPUS strategy then ensures efficient time integration by approximately balancing the space and time errors. The combination of these two approaches provides a new, fully automatic algorithm with both spatial and temporal error control.

Acknowledgements: Jane Lawson acknowledges the support of Shell Research Limited and the S.E.R.C. through a CASE studentship. Martin Berzins acknowledges the financial support of the Rensselaer Design Research Center while on leave from Leeds University.

References

[1] Babuska, I. and Luskin, M. (1981) An Adaptive Time Discretization Procedure for Parabolic Problems, ***Advances in Comp. Meths. for PDEs***, **IV**, 5-8.

[2] Babuska, I. and Rheinboldt, W.C. (1978) A Posteriori Error Estimates for the Finite Element Method, ***Int. J. for Num. Meths. in Engng.***, **12**, 1597-1615.

[3] Berzins, M. (1988) Global Error Estimation in the Method of Lines for Parabolic Equations, *SIAM J. Sci. Stat. Comput.*, **9**, 687-703.

[4] Berzins, M., Dew, P.M. and Furzeland, R.M. (1989) Developing PDE Software Using the Method of Lines and Differential Algebraic Integrators, *Appl. Numer. Maths.*, 5, 375-397.

[5] Bieterman, M. and Babuska, I. (1986) An Adaptive Method of Lines with Error Control for Parabolic Equations of the Reaction-Diffusion Type, *J. Comput. Phys.*, **63**, 33-66.

[6] Johnson, C. (1987). Numerical Solution of PDEs by the *Finite Element Method*, Cambridge University Press.

[7] Lawson, J., Berzins, M. and Dew, P.M. (1991) Balancing Space and Time Errors for Parabolic Equations. *SIAM J. Sci. Stat. Comput.*, ~~to appear~~. 12, 573-594.

[8] Lawson, J. (1989) Towards Error Control for the Numerical Solution of Parabolic Equations, Ph.D. Thesis, School of Computer Studies, University of Leeds.

[9] Schoñauer, W., Schnepf, E. and Raith, K. (1984) Experiences in Designing PDE Software with Self Adaptive Variable Stepsize / Order Difference methods, *Computing*, **5**, 227-242.

[10] Shampine, L.F. (1987) Tolerance Proportionality in ODE Codes, SMU Math. Rept. 87-8, Southern Methodist University, TX 75275, U.S.A..

[11] Skeel, R.D. and Berzins, M. (1990) Improving Routines for Solving Parabolic Equations in One Space Variable, *SIAM J. Sci. Stat. Comput.*, 11, 1-32.

[12] Stetter, H.J. (1976) Considerations Concerning a Theory for ODE Solvers, *Numerical Treatment of Differential Equations*, ed. by R. Bulirsch, R.D. Grigorieff and J. Schroder, Lecture Notes in Mathematics **631**, Springer Verlag, New York 188-200.