IMPLICIT SOLUTION OF HYPERBOLIC EQUATIONS WITH SPACE-TIME ADAPTIVITY

PER LÖTSTEDT[1], STEFAN SÖDERBERG[2], ALISON RAMAGE[3], and LINA HEMMINGSSON-FRÄNDÉN[4] *

[1] *Dept of Information Technology, Scientific Computing, Uppsala University, SE-75104 Uppsala, Sweden. email: perl@tdb.uu.se*

[2] *Dept of Information Technology, Scientific Computing, Uppsala University, SE-75104 Uppsala, Sweden. email: stefans@tdb.uu.se*

[3] *Dept of Mathematics, University of Strathclyde, Glasgow G1 1XH, Scotland. email: A.Ramage@strath.ac.uk*

[4] *Dept of Information Technology, Scientific Computing, Uppsala University, SE-75104 Uppsala, Sweden. email: lina@tdb.uu.se*

Abstract.

Adaptivity in space and time is introduced to control the error in the numerical solution of hyperbolic partial differential equations. The equations are discretised by a finite volume method in space and an implicit linear multistep method in time. The computational grid is refined in blocks. At the boundaries of the blocks, there may be jumps in the step size. Special treatment is needed there to ensure second order accuracy and stability. The local truncation error of the discretisation is estimated and is controlled by changing the step size and the time step. The global error is obtained by integration of the error equations. In the implicit scheme, the system of linear equations at each time step is solved iteratively by the GMRES method. Numerical examples executed on a parallel computer illustrate the method.

AMS subject classification: 65M50, 65N22.

Key words: Finite volume method, linear multistep method, error control, adaptivity, GMRES, parallel computation.

1 Introduction

Adaptivity simplifies the numerical solution of time-dependent partial differential equations (PDEs) for a user by automatically refining and coarsening the computational grid and selecting an appropriate time step. The discretisation error in the solution of the PDE depends on both the grid size and on the time step. By adapting the discretisation to the solution, it is possible to control the numerical errors so that they satisfy user-supplied tolerances. Furthermore, there is no waste of grid cells where the solution is smooth and there is no need for a dense grid.

*Financial support has been obtained from the Swedish National Network for Applied Mathematics and the Swedish Research Council for Engineering Science.

One major difficulty is that the regions in space and time where a fine resolution is necessary are not known in advance. Only an experienced user can distribute the cells in an economical way over the computational domain. In some parts of the time interval, a large time step may be satisfactory, but in other parts a small time step is necessary for an accurate solution. In addition, if the time step is constant it has to be chosen so that the most demanding parts are well resolved, resulting in too many time steps. Hence, both CPU-time and storage are saved by adaptive numerical methods.

In this paper, the space grid is partitioned into predetermined blocks and the grids are structured in each block, but the techniques can be extended to unstructured grids. The step size of the grid may have jumps at the block boundaries, but the difference in step size is at most a factor of two across a boundary. Block-wise adaptivity has the advantage of low overhead costs in both storage and computing time and it is easier to maintain second order accuracy in the whole domain. The PDEs are approximated by a finite volume method (FVM) in space and a linear multistep method (LMM) in time. The FVM is the usual centred scheme in the interior of a block with a second order upwind modification at the block boundaries with discontinuous cell sizes. This is necessary for stability. The LMM is the implicit backward differentiation method of second order. Thus, the overall discretisation is of second order on Cartesian grids with constant step sizes. Note, however, that the scheme is not conservative at block boundaries with jumps in the grid size. By enforcing strict conservation, one order of accuracy is lost.

An alternative to the LMM is a Runge-Kutta method (RK). In a LMM, we reuse old evaluations of the space derivatives from previous time steps. The old approximations of the derivatives satisfy an error tolerance and are expensive to compute. In an RK-method, the space discretisation is computed at each inner stage and may have to be changed at an inner stage of the time step. This introduces extra complexity in a code using RK time stepping, in particular if the method is implicit.

Stability and conservation at boundaries with a change in the step size have been studied in [1] and [2]. Also, methods for adapting structured grids where the grid has patches of refinement have been developed in [3], [6], [12], [20], [25], [27]. New cells are introduced in the original grid in [8], [10], [26], [29]. In [24] the original grid is distorted to better fit the features of the solution. In all of these references, either the steady state solution is computed or the time step is constant in time-dependent problems.

The numerical errors are estimated in space by comparing the discretisation on two different grids, and the local error in time is estimated by comparing two different time stepping schemes. If the spatial error is too large in a cell in a block, then all of the cells in that block are refined. If all cells in a block have a sufficiently small error, then the cells are coarsened in that block. The error control in time is similar to the practice used in the solution of ordinary differential equations (ODEs) [5],[15]. That is, the global error in the solution is obtained by integrating the error equation on a coarser grid.

In [8], [10], and [29], the adaptivity in space is based on features such as gradients in the solution and in [19] areas of refinement are detected by wavelets. An estimate of the local truncation error is determined by comparing the solutions computed with two different step sizes and time steps in [3] and [12]. The advantage with control of the local error is that the solution error can be estimated and the user has quantitative information about the accuracy of the solution. A parabolic PDE is solved in [4] with estimates of the temporal and spatial errors. In [11], tolerances on the global error in a finite element discretisation are satisfied by utilising the computed solution to the adjoint problem.

With an implicit method, a system of linear equations has to be solved at each time step. The system is solved using the GMRES method [30] with or without block preconditioning. The preconditioner is based on Toeplitz blocks with a certain structure that gives the preconditioned system a fast solver of Poisson type [16]. An efficient preconditioner is necessary when the time scales in the equations differ significantly. In the examples in this paper this is not the case, and the description and testing of the preconditioned solver can be found in a separate paper.

The adaptive method is implemented for execution in parallel. The blocks of the grid are distributed dynamically over the available processors for a good load balance and the data at the boundaries of neighbouring blocks on a different processor are communicated by message passing. Parallel implementations of adaptive methods for time-independent PDEs in flow problems are described in [9], [32].

In the next section, the space and time discretisation and the error estimates are described. The local error is of second order in all cells. Then the stability and conservation properties of the scheme are discussed in the third section. Section 4 deals with the iterative solution of the linear system of equations and the block preconditioning, and the parallel implementation is described in Section 5. One numerical experiment in the last section is chosen so that the analytic solution is known. The error estimates can then be compared to true errors in the numerical solution.

2 Discretisation

The hyperbolic differential equation in conservation form

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

is solved for $(x, y, t) \in \Omega \times [0, t_{end}]$ where $f = f(u)$ and $g = g(u)$ are the flux functions. The Jacobian matrices of f and g are

$$\frac{\partial f}{\partial u}=F(u),\ \frac{\partial g}{\partial u}=G(u).$$

This equation is discretised by a finite volume method in space and a linear multistep method in time on a structured grid, which is partitioned into p blocks. To simplify the communication of data between the blocks at the block boundaries, two rows of ghost cells are added there so that the grids in the blocks partly overlap each other.

2.1 Space discretisation

Let the area of the cell ω_{jk} be $|\omega_{jk}|$. Then the cell average of u computed by the FVM is defined by

$$u_{jk}(t) = \frac{1}{|\omega_{jk}|}\int_{\omega_{jk}} u(x,y,t)d\omega. \tag{2.2}$$

After integrating (2.1) over ω_{jk} using (2.2) and Gauss' theorem we have

$$\frac{\partial u_{jk}}{\partial t} + \frac{1}{|\omega_{jk}|}\int_{\partial\omega_{jk}} (fn_x + gn_y)dS = 0, \tag{2.3}$$

where (n_x, n_y) is the normal at the cell boundary $\partial\omega_{jk}$. The cells are such that the normal is constant at each cell face. The accuracy in space in a cell depends on how u is approximated on $\partial\omega_{jk}$ from the cell averages. Then f and g are computed on $\partial\omega_{jk}$.

We consider approximations that are of second order on Cartesian grids with constant step sizes Δx and Δy in the x and y directions. For curvilinear grids, the local errors are generally not of second order, but the coefficient in front of the first order term is often small. Let j and k be the indices in the x and y directions, respectively. On the face $(j+1/2,k)$ between the cells (j,k) and $(j+1,k)$, the usual choices for $u_{j+1/2,k}$ are

$$u_{j+1/2,k} = (u_{jk} + u_{j+1,k})/2, \tag{2.4}$$

yielding a centred difference approximation of $\partial u/\partial x$ when $f = u$, or the upwind approximation

$$u_{j+1/2,k} = (3u_{jk} - u_{j-1,k})/2 \;\text{ or }\; u_{j+1/2,k} = (3u_{j+1,k} - u_{j+2,k})/2. \tag{2.5}$$

The y direction is treated similarly. To investigate the accuracy of (2.4) and (2.5) consider the Taylor expansion around the lower left corner of cell (j,k) at $(0,0)$ with $\Delta x = \Delta y = h$. For (2.4) we have

$$\begin{aligned} h^{-1}\int_0^h (u_{j-1/2,k} - u(0,y))dy &= h^2 u_{xx}/6 + O(h^3), \\ h^{-1}\int_0^h (u_{j+1/2,k} - u(h,y))dy &= h^2 u_{xx}/6 + O(h^3), \end{aligned} \tag{2.6}$$

where the subscript x on u denotes differentiation and u_{xx} is evaluated at $(0,0)$. In the same manner we derive for (2.5)

$$\begin{aligned} h^{-1}\int_0^h (u_{j-1/2,k} - u(0,y))dy &= h^2 u_{xx}/3 + O(h^3), \\ h^{-1}\int_0^h (u_{j+1/2,k} - u(h,y))dy &= h^2 u_{xx}/3 + O(h^3). \end{aligned} \tag{2.7}$$

The average u_{jk} is well approximated by u at the midpoint

$$u_{jk} - u(h/2,h/2) = h^2(u_{xx} + u_{yy})/24 + O(h^3).$$

The accuracy of the discretisation of the integral in the x direction in (2.3) is obtained as above. Let the first term in the right-hand side in (2.6) and (2.7) be written $c_u h^\nu$. Expansion of f gives

$$
\begin{aligned}
&h^{-1}\int_0^h (f(u_{j-1/2,k}) - f(u(0,y)))dy \\
&= h^{-1}\int_0^h \int_0^1 F(u(0,y) + \zeta\cdot(u_{j-1/2,k} - u(0,y)))\, d\zeta \cdot (u_{j-1/2,k} - u(0,y))\, dy \\
&= c_f h^{-1}\int_0^h (u_{j-1/2,k} - u(0,y))dy + O(h^{\nu+1}) = c_f c_u h^\nu + O(h^{\nu+1}), \\
&h^{-1}\int_0^h (f(u_{j+1/2,k}) - f(u(h,y)))dy \\
&= c_f h^{-1}\int_0^h (u_{j+1/2,k} - u(h,y))dy + O(h^{\nu+1}) = c_f c_u h^\nu + O(h^{\nu+1}).
\end{aligned}
\tag{2.8}
$$

Here, the parameter c_f is $F(u)$ evaluated at $u(0,0)$ and ν is the order of the error in $u_{j\pm1/2,k}$. It then follows from (2.3) and (2.8) that

$$
\begin{aligned}
h^{-2}\int_{\omega_{jk}} \frac{\partial f}{\partial x} d\omega &= h^{-2}\int_0^h (f(u(h,y)) - f(u(0,y)))dy \\
&= h^{-1}(f(u_{j+1/2,k}) - f(u_{j-1/2,k})) + O(h^\nu).
\end{aligned}
\tag{2.9}
$$

It is important that the term of $O(h^\nu)$ in (2.8) is cancelled when the fluxes at $(j+1/2,k)$ and $(j-1/2,k)$ are subtracted from each other. Otherwise, the local order of accuracy will be only $\nu - 1$. This is the case if for example $u_{j-1/2,k}$ is determined by (2.4) and $u_{j+1/2,k}$ by (2.5). The accuracy of the approximation of the derivative in the y direction is derived in the same manner.

2.2 Interpolation at boundaries

The computational grid is partitioned into blocks when it is generated and the block structure remains the same throughout the simulation. Recall that each block has two rows of ghost cells overlapping the adjacent block. The solution in these cells is needed to compute the space discretisation in the cells at the block boundaries. The grid is always refined or coarsened in all cells in a block. In this way, the administration of the grid patches is simplified, particularly for time-dependent problems. The data structure is less complicated than for the case when each cell can be refined or when the refined patches can be located arbitrarily. It is also easier to achieve second order accuracy in all cells.

If the cells are of the same size on both sides of a block boundary, then the variables in the ghost cells are simply copied from the adjacent block. If there is a jump in the step size across a boundary, however, interpolation is necessary. For second order accuracy of the FVM on a Cartesian grid, the values in the ghost cells are such that the difference in (2.6) and (2.7) introduced by the interpolation is of $O(h^3)$.

The values in the ghost cells of the fine block are interpolated using only the data in the coarse block. Following the notation in Figure 2.1, for the case $\Delta x = \Delta y = h$ the interpolation is given by

$$
\begin{aligned}
u_3 &= \frac{21}{16}u_E + \frac{3}{16}u_B - \frac{7}{16}u_D - \frac{1}{16}u_A - \frac{1}{8}u_I + \frac{1}{8}u_C, \\
u_1 &= \frac{11}{16}u_E + \frac{1}{16}u_B + \frac{7}{16}u_D + \frac{1}{16}u_A - \frac{1}{8}u_I - \frac{1}{8}u_C,
\end{aligned}
\tag{2.10}
$$

in the fine block ghost cells 1 and 3. The values in cells 2 and 4 are obtained by replacing A by H and interchanging B and I in (2.10). In the coarse ghost cells F and G we take

$$\begin{aligned} u_F &= (u_5 + u_6 + u_7 + u_8)/4, \\ u_G &= (u_9 + u_{10} + u_{11} + u_{12})/4. \end{aligned} \tag{2.11}$$

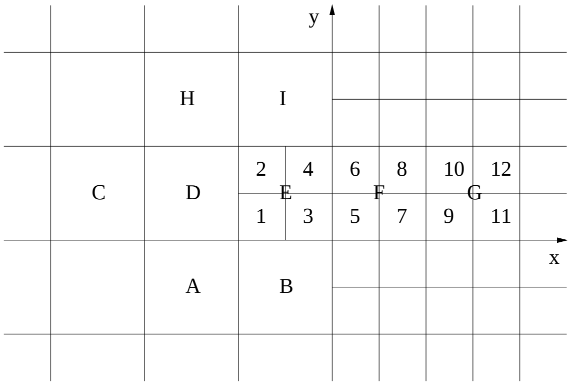

Figure 2.1: A block boundary with ghost cells and a jump in the step size at $x = 0$.

If $u_j, j = 5 \dots 12$, are the exact cell averages, then u_F and u_G are also the exact averages in the coarse cells F and G. The interpolated values in (2.10) are of $O(h^3)$. Thus, the error in the approximation of $u(0, y)$ and $u(h, y)$ in (2.6) and (2.7) is of the same form as the interpolated data and the discretisation (2.9) is still of second order. This particular interpolation in combination with the upwind scheme at the block boundaries gives a stable space discretisation (see Section 3). If the interpolation is of $O(h^2)$, then the perturbation at the cell face $x = 0$ is of $O(h^2)$ and the local error in (2.9) is of $O(h)$, which may be acceptable for the part of the solution propagated along characteristics across the boundary. It is shown in [14] that the accuracy is of $O(h^2)$ in the interior of the block even if it is of $O(h)$ at a boundary. For the part of the solution travelling along the boundary, the accuracy will only be of $O(h)$, cf. [13].

The simplest interpolation is to let

$$u_1 = u_3 = u_E. \tag{2.12}$$

The order of accuracy in u_1 and u_3 is 1 and the approximation of $\partial u/\partial x$ in cell 5 in Figure 2.1 with (2.4) is not consistent:

$$(2h)^{-1}(u_7 - u_3) = 5u_x/4 + O(h). \tag{2.13}$$

In cells (j, k) with at least one face on a block boundary with a jump in the step size across that boundary, the space derivatives are approximated by an upwind calculation of u_{jk} (2.5). In all other cells, the centred approximation (2.4) is used. The change of stencil at the boundary is necessary for stability there. The reason why the centred scheme is preferred in the interior is the smaller local error in (2.6) compared to (2.7). The upwind approximation in a boundary cell (j,k) can be interpreted as a centred approximation with a different interpolated value $u^*_{j-1,k}$ in the ghost cell $(j-1,k)$ as follows. Let E be the shift operator defined by $Eu_{jk}=u_{j+1,k}$. Then the difference stencils for u_x in (j,k) use the centred scheme and $u^*_{j-1,k}$ in the ghost cell and the upwind scheme and $u_{j-1,k}$ in the ghost cell:

$$h^{-1}(0.5u_{j+1,k}-0.5u^*_{j-1,k})=h^{-1}(1.5u_{jk}-2u_{j-1,k}+0.5u_{j-2,k}).$$

Thus,

$$u^*_{j-1,k}=u_{j-1,k}+(1-E^{-1})^3Eu_{jk},$$

and the difference is of $O(h^3)$.

2.3 Error equation

A differential equation approximately satisfied by the error e in the solution

$$\frac{\partial(u+e)}{\partial t}+\frac{\partial f(u+e)}{\partial x}+\frac{\partial g(u+e)}{\partial y}=\tau_t+\tau_s,$$

is derived in [13]. The error is assumed to be a smooth function in space and time. Denote the truncation errors when the analytic solution u is inserted into the difference formulas by $-\tau_s$ for the approximation of the space operator and $-\tau_t$ for the approximation of the time derivative. If f and g are smooth, using a Taylor expansion around u (ignoring higher order terms in e) gives the error equation for (2.1) as

$$\frac{\partial e}{\partial t}+F(u)\frac{\partial e}{\partial x}+G(u)\frac{\partial e}{\partial y}=\tau_t+\tau_s. \tag{2.14}$$

The linear equation (2.14) is discretised by the same methods as (2.1). Estimates of τ_s and τ_t are computed at discrete points in space and time, then (2.14) is solved on the next coarser grid in each block at every second time step. Hence, the work in 2D for this extra error information is about 1/8 of the work to solve (2.1).

2.4 Space discretisation error

The spatial error τ_s is estimated as follows. Let $\mathcal{F}_{jk}$ be the integral in (2.3) and let $\Phi_{jk}(u_{jk})$ be its finite volume approximation in cell (j,k). The truncation error is then

$$\tau_{sjk}=\mathcal{F}_{jk}-\Phi_{jk}(u_{jk}). \tag{2.15}$$

Assume that the cell ω_{jk} in a coarse grid with step size h consists of the q cells $\omega_{jkl}, l=1\ldots q$, with step sizes h/r in a fine grid and that the discretisation has order ν (in the examples in Section 6, $q = 4$, $\nu = 2$ and $r = 2$). It is shown in [13] that in the cell (j, k) and its q subcells,

$$\begin{aligned}\sigma &= \textstyle\sum_{l=1}^{q} \kappa_l \Phi_{jkl}(u_{jkl}) - \Phi_{jk}(\sum_{l=1}^{q} \kappa_l u_{jkl}),\\ \tau_{sjkl} &= \sigma/(r^\nu - 1), l = 1 \dots q, \tau_{sjk} = r^\nu \sigma/(r^\nu - 1),\end{aligned} \tag{2.16}$$

where κ_l is the quotient of the areas of the cells

$$\kappa_l = |\omega_{jkl}|/|\omega_{jk}|.$$

The local error is computed by comparing the restriction of the residual on the fine grid with the coarse grid residual evaluated at the restriction of the fine grid solution. An alternative would be to compare with a third or fourth order method. However, to implement such a scheme requires some effort and since the local error is controlled by a change in step size, it is a natural test to see the effect of such a change by computing the space discretisation for two different resolutions. The local error in time is estimated in the next section.

2.5 Time discretisation

The approximation of time derivatives is made by the backward differentiation formula of order two (BDF-2), see [15]. The time step is the same in every block: BDF-2 is an A-stable two step method and the coefficients of the scheme have to be modified if there is a change in the length of the time step. An advantage with the implicit method is that the time steps have to be adjusted only with respect to the accuracy. An explicit method often has to be implemented with different time steps in different blocks. Otherwise, the time integration would be too inefficient with the time step restricted for stability by the smallest step size in all blocks. The disadvantage with BDF-2 is having to solve a system of equations at every step. This question is addressed in Section 4.

We want to approximate u^n by $\Phi(u^n)$ and the known solutions u^{n-1} and u^{n-2} at t_{n-1} and t_{n-2}. The time step is defined by $\Delta t_i = t_i - t_{i-1}$. Using a Taylor expansion of u at t_n and cancelling terms of order less than 3 in the time step, we obtain a second order approximation with variable coefficients given by

$$\begin{aligned}&\alpha_0^n u^n = \Delta t_n \Phi(u^n) - \alpha_1^n u^{n-1} - \alpha_2^n u^{n-2},\\ &\theta^n = \Delta t_n/\Delta t_{n-1},\\ &\alpha_0^n = (1 + 2\theta^n)/(1 + \theta^n),\ \alpha_1^n = -(1 + \theta^n),\ \alpha_2^n = (\theta^n)^2/(1 + \theta^n).\end{aligned} \tag{2.17}$$

With $\Delta t = \max(\Delta t_{n-1}, \Delta t_n)$ the local error is

$$\begin{aligned}u(t_n) - u^n &= c_I(\theta^n)\Delta t_n^3 u''' + O(\Delta t^4),\\ c_I(\theta^n) &= -(1 + \theta^n)^2/(6\theta^n(1 + 2\theta^n)).\end{aligned} \tag{2.18}$$

If $\Delta t_{n-1} = \Delta t_n$, then we have the familiar coefficients (cf. [15])

$$\alpha_0 = 3/2,\ \alpha_1 = -2,\ \alpha_2 = 1/2,\ c_I = -2/9.$$

The leading term in the local error in time in (2.18) is $c_I(\theta^n)\Delta t_n^3 u'''$. It is estimated by comparing u^n computed by BDF-2 in (2.17) with $\tilde{u}^n$ determined by an explicit two step method with a different c_I but still using u^{n-1} and u^{n-2} ('Milne's device'). The difference $u^n - \tilde{u}^n$ gives an estimate of the leading term of the local error. The solution $\tilde{u}^n$ also serves as the initial guess in the iterative solution of (2.17). Another possibility would be to compute u^n with two different time steps and then compare the results.

Here we choose the following explicit method as predictor:

$$\begin{aligned} &\tilde{\alpha}_0^n \tilde{u}^n = \Delta t_n \Phi(u^{n-1}) - \tilde{\alpha}_1^n u^{n-1} - \tilde{\alpha}_2^n u^{n-2}, \\ &\tilde{\alpha}_0^n = 1/(1+\theta^n),\ \tilde{\alpha}_1^n = \theta^n - 1,\ \tilde{\alpha}_2^n = -(\theta^n)^2/(1+\theta^n), \end{aligned} \tag{2.19}$$

with the local error

$$\begin{aligned} &u(t_n) - \tilde{u}^n = c_P(\theta^n)\Delta t_n^3 u''' + O(\Delta t^4), \\ &c_P(\theta^n) = (1 + 1/\theta^n)/6. \end{aligned} \tag{2.20}$$

If $\partial\Phi/\partial u$ is of $O(1)$, then we infer from (2.17), (2.18), and (2.20) that the leading term in the discretisation error in the BDF-2 approximation of u' in (2.14) is

$$\tau_t^n = -\alpha_0^n c_I \Delta t_n^2 u''' = \alpha_0^n c_I (u^n - \tilde{u}^n)/(\Delta t_n (c_I - c_P)). \tag{2.21}$$

The first step from $t_0 = 0$ to $t_1 = \Delta t_1$ is computed by BDF-1 or the backward Euler method with $\alpha_0^1 = 1$, $\alpha_1^1 = -1$ and $\alpha_2^1 = 0$.

3 Stability and conservation

In this section we consider the grid in Figure 3.1 and investigate the stability and conservation properties of the FVM and LMM discretisation outlined in Section 2.

3.1 Stability

For stability, we study the special case of (2.1) given by

$$\frac{\partial u}{\partial t} + \sin\psi \frac{\partial u}{\partial x} + \cos\psi \frac{\partial u}{\partial y} = 0,\ \psi \in [0, \pi], \tag{3.1}$$

on a domain with one coarse and one fine block as in Figure 3.1 with indices $j = -J : J$ and $k = -K : K$. The boundaries are periodic so that the upper face of cell (j, K) is the same as the lower face of $(j, -K)$ and the right faces of cells (J, k) and $(J, k+1)$ coincide with the left face of cell $(-J, k)$. The block boundary is located between $j = 0$ and 1.

Equation (3.1) is discretised in space by the scheme in Section 2 to obtain

$$\frac{\partial u}{\partial t} + \sin\psi H_F u + \cos\psi H_G u = 0 \tag{3.2}$$

where H_F and H_G are matrices. Since $\sin\psi \geq 0$, the flow crosses the block boundary from the coarse to the fine grid at $j = 1/2$ and from the fine to the coarse grid at the periodic x boundary. The stability analysis is complicated (see [1] for 1D examples) but in numerical calculations, the eigenvalues of the matrix

$$H(\psi) = \sin\psi H_F + \cos\psi H_G$$

satisfy

$$-Re\lambda(H(\psi)) \le c_\lambda, c_\lambda \approx 10^{-14},$$

for different values of $\psi \in [0, \pi]$ and J and K. The time integration is stable if $c_\lambda \le 0$ since BDF-2 is A-stable.

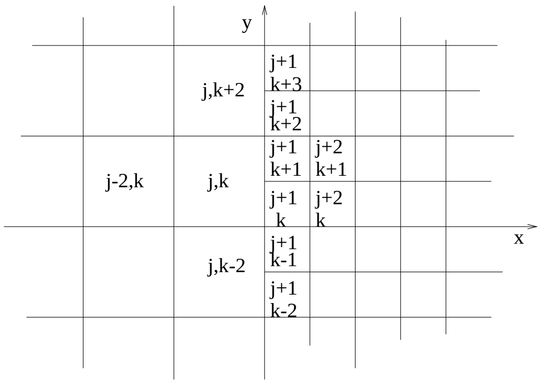

Figure 3.1: A block boundary with a jump in the step size at $x = 0$ showing the indexing of the cells.

Other alternatives have been investigated, but at the block boundaries with jumps in the cell size only one-sided interpolation as in (2.10) and (2.11) in combination with upwinding is stable. An intuitive explanation for this is that the interpolated values in cells $1, 2, 3, 4$ in Figure 2.1 are computed from upwind data and an upwind discretisation introduces the necessary dissipation.

For the simple interpolation (2.12) and $u_F = 0.5(u_5 + u_6)$ in Figure 2.1 with the centred scheme (2.4), introduce weights ξ_{jk} proportional to the cell areas such that $\xi_{jk} = 4$ if $j \le 0$ and 1 if $j > 0$. Now multiply (3.2) by $\Xi = \mathrm{diag}(\xi_{jk})$ to obtain

$$\frac{\partial}{\partial t}(\Xi u) + \sin\psi \Xi H_F u + \cos\psi \Xi H_G u = 0. \tag{3.3}$$

One can show that $(\Xi H_F)^T = -\Xi H_F$ and $(\Xi H_G)^T = -\Xi H_G$. Consequently, $(\Xi H(\psi))^T = -\Xi H(\psi)$ and in the norm

$$\|v\|_\Xi^2 = v^T \Xi v,$$

the solution of the ODE (3.3) is stable. It follows from [7] that the BDF-2 discretisation is also stable in a certain norm.

3.2 Conservation

The discretisation of (2.1) in cell (j,k) for $j>0$ in Figure 3.1 at $t=t_n$ is given by

$$\begin{aligned} \Delta t_n^{-1}\sum_{i=0}^2 \alpha_i^n u_{jk}^{n-i} \quad & +h^{-1}(f_{j+1/2,k}^n - f_{j-1/2,k}^n) \\ & +h^{-1}(g_{j,k+1/2}^n - g_{j,k-1/2}^n) = 0. \end{aligned} \tag{3.4}$$

The numerical flux function $f_{j+1/2,k}^n$ is evaluated at $t=t_n$ at the cell face $j+1/2$ between the cells (j,k) and $(j+1,k)$. It is assumed that $f_{j+1/2,k}^n$ is consistent so that

$$f_{j+1/2,k}^n = \phi(u_{j-m+1,k}^n, u_{j-m+2,k}^n \dots u_{j+m,k}^n),$$

and $\phi(u,u,\dots,u)$ equals the analytical flux function $f(u)$ in (2.1) (see [23]). The definition of $g_{j,k+1/2}^n$ is similar. For $j\le 0$ in (3.4), $j\pm 1/2$ and $k\pm 1/2$ are replaced by $j\pm 1$ and $k\pm 1$, respectively. If $\alpha_0^n=1$, $\alpha_1^n=-1$, and $\alpha_2^n=0$, then (3.4) is written in conservative form.

Let φ be a function in C_0^∞ and consider the Cauchy problem in $\mathbf{R}^2\times[0,\infty)$. Multiply (3.4) by φ, the cell area and the time step and take the sum over all cells and time steps from $t=0$ until ∞. It is shown in [23] that if the scheme (3.4) is in conservative form, $h,\Delta t\to 0$ and u_{jk}^n converges boundedly to u almost everywhere, then the sum converges to the weak form of (2.1)

$$-\int_0^\infty \int\int (\varphi_t u + \varphi_x f + \varphi_y g)dxdydt - \int\int \varphi^0 u^0 dxdy = 0, \tag{3.5}$$

satisfied by u, where the subscripts on φ denote differentiation and the superscript 0 the initial condition at $t=0$. We will now show that under certain conditions this is also true for BDF-2 with variable time steps and an inner block boundary as in Figure 3.1.

At the block boundary at $j=1/2$ in Figure 3.1, the flux in the x direction in the cell columns $j=0$ and $j=1$ is $\tilde{f}_{jk}$ and in all other cells f_{jk}.

THEOREM 3.1. *Suppose that equation (3.4) is solved by BDF-2 (2.17) in steps $n\ge 2$ and by BDF-1 in step 1. Assume that the approximations are consistent and that*

1. $\theta^n = \theta^{n-1} + c_\theta \Delta t_n^2$ *with* $\theta^1 = 1$,
2. $f_{-1,k} = \tilde{f}_{-1,k} + \tilde{c}_0 h,\ f_{3/2,k} = \tilde{f}_{3/2,k} + \tilde{c}_1 h$,
3. $\tilde{f}_{1k} = 0.5(\tilde{f}_{1/2,k} + \tilde{f}_{1/2,k+1}) + c_f h$,
4. $u_{jk}^1 = u_{jk}^0 + c_u \Delta t_1$,

where $c_\theta, \tilde{c}_0, \tilde{c}_1, c_f, c_u$ depend smoothly on x, y, and t and are bounded independently of h and Δt_n. Multiply (3.4) by φ and sum over all cells and time steps from $n=1$ and let $\Delta t = \max \Delta t_n$. Assume that u_{jk}^n converges boundedly to u almost everywhere. As $h,\Delta t\to 0$ the sum converges to (3.5), and u is a weak solution of (2.1).

PROOF. Let $\tilde{\sum}$ denote summation with increment 2 and split the sum into 5 parts

$$\begin{aligned}
S_1 &= \tilde{\sum}_{j=-\infty}^{0}\tilde{\sum}_{k=-\infty}^{\infty}|\omega_{jk}|S_{jk} + \sum_{j=1}^{\infty}\sum_{k=-\infty}^{\infty}|\omega_{jk}|S_{jk},\\
S_{jk} &= \sum_{n=2}^{\infty}\sum_{i=0}^{2}\alpha_i^n\varphi_{jk}^n u_{jk}^{n-i} + \varphi_{jk}^1(\alpha_0^1 u_{jk}^1 + \alpha_1^1 u_{jk}^0),\\
S_2 &= \tilde{\sum}_{k=-\infty}^{\infty}\sum_{n=1}^{\infty}\Delta t_n(\tilde{\sum}_{j=-\infty}^{-2}|\omega_{jk}|(2h)^{-1}\varphi_{jk}^n(f_{j+1,k}^n - f_{j-1,k}^n)\\
&\quad +|\omega_{0k}|(2h)^{-1}\varphi_{0k}^n(\tilde{f}_{1k}^n - \tilde{f}_{-1,k}^n)),\\
S_3 &= \sum_{k=-\infty}^{\infty}\sum_{n=1}^{\infty}\Delta t_n(\sum_{j=2}^{\infty}|\omega_{jk}|h^{-1}\varphi_{jk}^n(f_{j+1/2,k}^n - f_{j-1/2,k}^n)\\
&\quad +|\omega_{1k}|h^{-1}\varphi_{1k}^n(\tilde{f}_{3/2,k}^n - \tilde{f}_{1/2,k}^n)),\\
S_4 &= \tilde{\sum}_{j=-\infty}^{0}\sum_{n=1}^{\infty}\Delta t_n\tilde{\sum}_{k=-\infty}^{\infty}|\omega_{jk}|(2h)^{-1}\varphi_{jk}^n(g_{j,k+1}^n - g_{j,k-1}^n),\\
S_5 &= \sum_{j=1}^{\infty}\sum_{n=1}^{\infty}\Delta t_n\sum_{k=-\infty}^{\infty}|\omega_{jk}|h^{-1}\varphi_{jk}^n(g_{j,k+1/2}^n - g_{j,k-1/2}^n).
\end{aligned}$$

After partial summation in S_2 and S_3 we have

$$\begin{aligned}
S_2 = &\ -\tilde{\sum}_{k=-\infty}^{\infty}\sum_{n=1}^{\infty}\Delta t_n(\tilde{\sum}_{j=-\infty}^{0}|\omega_{jk}|(2h)^{-1}(\varphi_{jk}^n - \varphi_{j-2,k}^n)f_{j-1,k}^n\\
&\ -2h\varphi_{0k}^n(\tilde{f}_{1k}^n + f_{-1,k}^n - \tilde{f}_{-1,k}^n)),\\
S_3 = &\ -\sum_{k=-\infty}^{\infty}\sum_{n=1}^{\infty}\Delta t_n(\sum_{j=1}^{\infty}|\omega_{jk}|h^{-1}(\varphi_{j+1,k}^n - \varphi_{j,k}^n)f_{j+1/2,k}^n\\
&\ +h\varphi_{1k}^n(\tilde{f}_{1/2,k}^n + f_{3/2,k}^n - \tilde{f}_{3/2,k}^n)).
\end{aligned}$$

If we let $h, \Delta t \to 0$ and use the consistency of f, then we obtain

$$\begin{aligned}
S_2 + S_3 &= \int_{-\infty}^{\infty}\int_0^{\infty}\int_{-\infty}^{\infty} -\varphi_x f\,dx\,dt\,dy\\
&\quad + \sum_{n=1}^{\infty}\Delta t_n(\tilde{\sum}_{k=-\infty}^{\infty}h(2\varphi_{0k}^n\tilde{f}_{1k}^n - (\varphi_{1k}^n\tilde{f}_{1/2,k}^n + \varphi_{1,k+1}^n\tilde{f}_{1/2,k+1}^n))\\
&\quad +h(2\varphi_{0k}^n(f_{-1,k}^n - \tilde{f}_{-1,k}^n) - \varphi_{1k}^n(f_{3/2,k}^n - \tilde{f}_{3/2,k}^n))),
\end{aligned}$$

where the last sum vanishes thanks to assumptions 2 and 3 and the smoothness of φ. In the same manner,

$$S_4 + S_5 \to \int_0^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} -\varphi_y g\,dy\,dx\,dt.$$

The coefficients α_i^n in (2.17) are smooth functions of θ^n when $\theta^n > 1$. Thus, with the derivative α_i' evaluated between θ^{n-1} and θ^n and by assumption 1,

$$\alpha_i^n = \alpha_i^{n-1} + \alpha_i'(\theta^n - \theta^{n-1}) = \alpha_i^{n-1} + O(\Delta t_n^2),\ n \geq 2.$$

In the first sum S_1 we have, by assumptions 1 and 4, the boundedness of u and the smoothness of φ so that for each cell (j,k) as $\Delta t \to 0$

$$\begin{aligned}
&S_{jk} = \sum_{n=2}^{\infty}\Delta t_n(\Delta t_n^{-1}\sum_{i=0}^{2}\alpha_i^n\varphi^n u^{n-i}) + \Delta t_1(\Delta t_1^{-1}(\varphi^1(\alpha_0^1 u^1 + \alpha_1^1 u^0)))\\
&= \sum_{i=0}^{2}\sum_{n=2}^{\infty}\alpha_i^{n+i}\varphi^{n+i}u^n + \alpha_1^2\varphi^2 u^1 + \alpha_2^2\varphi^2 u^0 + \alpha_2^3\varphi^3 u^1\\
&+\alpha_0^1\varphi^1 u^1 + \alpha_1^1\varphi^1 u^0 = \sum_{i=0}^{2}\sum_{n=2}^{\infty}(\alpha_i^n\varphi^{n+i}u^n + O(\Delta t_n^2))\\
&+u^1(\alpha_0^2\varphi^1 + \alpha_1^2\varphi^2 + \alpha_2^2\varphi^3) + u^0(\alpha_2^2\varphi^2 + (\alpha_1^1 + \alpha_0^1 - \alpha_0^2)\varphi^1) + O(\Delta t_1)\\
&\to \sum_{n=1}^{\infty}\Delta t_n(-\varphi_t u)^n - u^0\varphi^0,
\end{aligned}$$

since $\alpha_0^2 = 3/2 + O(\Delta t_1^2)$ and $\alpha_2^2 = 1/2 + O(\Delta t_1^2)$. The sum $\sum_{i=0}^{2} \alpha_i^n \varphi^{n+i}$ converges to $-\Delta t_n \varphi_t$ and, since φ is smooth, the sum multiplied by u^0 is $-\varphi^0$. We conclude that

$$S_1 \to \int_0^\infty \int_{-\infty}^\infty \int_{-\infty}^\infty -\varphi_t u \, dx \, dy \, dt - \int_{-\infty}^\infty \int_{-\infty}^\infty u^0 \varphi^0 dx \, dy,$$

and the theorem follows from summation of S_1, S_2, S_3, S_4 and S_5. □

REMARK 3.1. The first part of assumption 1 can be replaced by

$$\theta^n = 1 + c_t^n \Delta t_n,$$

since

$$\theta^n - \theta^{n-1} = 1 - c_t^n \Delta t_n - 1 - c_t^{n-1} \Delta t_{n-1} = O(\Delta t_n^2).$$

The first assumption in the theorem is fulfilled if the quotient between the successive time steps is changed slowly. It follows from (2.8), the centred (f) and upwind ($\tilde{f}$) discretisations in (2.6) and (2.7) and the interpolation (2.10) and (2.11) that assumption 2 is satisfied by our treatment of block boundaries with jumps in the step size. Similar conditions are derived in [2]. The last assumption excludes an initial discontinuity in u and is needed because of the change of method between $n = 1$ and $n = 2$. The results in the theorem can be generalised to a fixed number of block boundaries as we have in our algorithm with assumptions 2 and 3 fulfilled at every boundary.

Strict conservation so that $c_f = 0$ in assumption 3 is not required for convergence of the sum to the weak form (as observed also in [2]). To ensure the correct shock speeds in compressible flow problems, strict conservation is usually enforced in the numerical scheme. Slowly moving shocks do not pass an interface between overlapping grids if the interpolation is not strictly conservative [28]. Unfortunately, strict conservation implies a loss of accuracy in our boundary treatment as is shown in the next proposition.

PROPOSITION 3.2. *Assume that the numerical fluxes in Figure 2.1 at the block boundary are computed to ν:th order accuracy as in (2.8) with $\nu \geq 1$. If strict conservation is enforced at the boundary, then the accuracy is lowered to order $\nu - 1$.*

PROOF. Consider the block boundary in Figure 2.1. The fluxes between the cells E and F, 3 and 5, and 4 and 6 are, according to (2.8),

$$\begin{aligned}
f_{EF} &= (2h)^{-1} \textstyle\int_0^{2h} f(u(0,y))dy + 2^\nu c_f c_u h^\nu + O(h^{\nu+1}), \\
f_{35} &= h^{-1} \textstyle\int_0^{h} f(u(0,y))dy + c_f c_u h^\nu + O(h^{\nu+1}), \\
f_{46} &= h^{-1} \textstyle\int_h^{2h} f(u(0,y))dy + c_f c_u h^\nu + O(h^{\nu+1}).
\end{aligned}$$

For strict conservation we need a correction Δf such that

$$f_{EF} + \Delta f = (f_{35} + f_{46})/2.$$

Thus,

$$\Delta f = (1 - 2^\nu) c_f c_u h^\nu + O(h^{\nu+1}).$$

Consequently, a change of order $h^{\nu-1}$ is introduced in (2.9) unless $\nu = 0$ as in (2.13). □

The method in Section 2 is second order accurate everywhere but it is not strictly conservative at block boundaries with jumps in the step size. If a shock crosses a boundary, the grid refinement algorithm will detect that and both blocks will have the same step size there. The numerical fluxes at the boundary are computed by a centred approximation in both blocks and we have strict conservation. In assumptions 2 and 3 in Theorem 3.1 we have $c_f = \tilde{c}_0 = \tilde{c}_1 = 0$. The method defined by (2.12) is strictly conservative and stable for (3.1) but inaccurate.

4 Iterative method

The discretisation defined in Section 2 leads to a large, sparse, linear system of equations to solve at each time step of the form

$$\begin{pmatrix} A & B \\ C & I \end{pmatrix} \begin{pmatrix} u^I \\ u^G \end{pmatrix}^{n+1} = \begin{pmatrix} b^I \\ b^G \end{pmatrix}, \tag{4.1}$$

where u^I denotes the unknowns within the blocks and u^G the unknowns in the ghost cells. The coefficient matrix A is given by $A = \mathrm{diag}(A_1, \ldots, A_p)$ where A_k is the local coefficient matrix for the unknowns in block k. The matrix B describes the coupling between u^I and u^G while C represents the interpolation. We will solve (4.1) by reducing the system of equations to a smaller one for u^I using block-Gaussian elimination, yielding

$$(A - BC)u^I = b^I - Bb^G. \tag{4.2}$$

Equation (4.2) is solved using restarted GMRES [30] with restart length l. Define the residual $r^{(i)}$ in iteration i by

$$r^{(i)} = b^I - Bb^G - (A - BC)u^{I,(i)}, \tag{4.3}$$

where $u^{I,(i)}$ denotes the approximation to u^I in iteration i. Using GMRES, the following will hold for each iteration:

$$\|r^{(i)}\|_2 = \min_{p_i \in \mathcal{P}_i, p_i(0)=1} \|p_i(A - BC) r^{(0)}\|_2,$$

where $\mathcal{P}_i$ is the set of all polynomials of degree i. If $A - BC$ is diagonalisable we obtain [30]

$$\frac{\|r^{(i)}\|_2}{\|r^{(0)}\|_2} \le \mathrm{cond}_2(W_{ABC}) \min_{p_i \in \mathcal{P}_i, p_i(0)=1} \max_{1 \le l \le n} |p_i(\lambda_l)|,$$

where W_{ABC} is the eigenvector-matrix of $A - BC$ and $\mathrm{cond}_2(\cdot)$ is the condition number of the matrix in the Euclidean norm. Hence we can hope for good convergence if the eigenvalues are clustered in a few dense clusters and W_{A-BC} is well conditioned. In [18] it is shown that for relatively small off-diagonal elements, this iteration will converge rapidly even without preconditioning. The problems that we will solve in our examples are such that the off-diagonal elements are of the same order as the diagonal elements.

For problems involving different timescales, for example low Mach number flows, the off-diagonal elements will be of order $\Delta t/(hM)$ where M is a small Mach number, while the diagonal will be of order 1. This means that we will have to use a preconditioner to ensure fast convergence. We will use a semi-Toeplitz preconditioner [17] for (4.2) for such problems in a forthcoming paper.

5 Adaptive algorithm

The adaptive algorithm is implemented for parallel execution. The user must specify an error tolerance: the discretisation errors τ_s and τ_t are kept below that bound. The error equations are integrated on a coarser grid to obtain a global error estimate.

The grid in a particular block is refined if the absolute value of the estimated error in one of the variables in at least one of the cells is greater than ϵ. It is coarsened if all variables in all cells in the block have an error less than 0.1ϵ. If the grid is changed, then the current solution is interpolated onto the cells in the new grid structure. The interpolation from a fine grid to a coarser grid is made by area-weighted averaging. The values in a coarse grid are transferred to a fine grid by bilinear interpolation.

In order to reduce the occurrence of reflections at the block boundaries, the refinement procedure utilises the following look-ahead strategy: if an adjacent block is at a coarser level and if the error at the corresponding block boundary is larger than a tolerance 0.6ϵ, the adjacent block is refined.

Let $\eta_t = \max_{jk} |\tau_{tjk}|$ be the maximum local error in time estimated by (2.21). For every time step n, compute the quotient

$$\zeta = \sqrt{\epsilon/\eta_t}. \tag{5.1}$$

If $0.98 \le \zeta \le 1.15$, then accept the step and let $\Delta t_{n+1} = \Delta t_n$. If $\zeta \le 0.98$, then the step to t_n is rejected and recomputed with $\Delta t_n = 0.9\zeta\Delta t_n$. If $\zeta > 1.15$, then the step is accepted and the time step increased with $\Delta t_{n+1} = 1.1\Delta t_n$. In this way, the time step does not vary too abruptly and assumption 1 in Theorem 3.1 can be satisfied.

In codes for solution of ODEs (see for example [5]), time steps are usually kept constant in implicit methods to avoid frequent factorisations of the system matrix in the Newton iterations. There is no such constraint here, as the equations are solved iteratively.

In the parallel implementation, a set of blocks is processed by each processor. MPI is used for exchange of data in the ghost cells between blocks belonging to different processors. The blocks are distributed dynamically on the processors for a good load balance by the diffusion algorithm in ParMetis [21]. The performance of ParMetis and a comparison with other approaches for the space adaptation algorithm are found in [31].

One time step from t_{n-1} to t_n is performed as follows:

(5.2)

1. Compute the spatial error τ_s in the fine grid cells as in (2.16).
2. Check if the grid should be refined or coarsened at t_{n-1} based on τ_s and, if necessary, interpolate the solutions at t_{n-2} and t_{n-1} onto the new grid.
3. Compute $\Phi(u^{n-1})$ and take a predictor step with (2.19).
4. Compute the corrector solution with the implicit method (2.17) and GMRES iterations.
5. Compute the temporal error as in (2.21).
6. Check if the time step must be decreased and the time step recomputed or if the step is accepted and a longer time step is possible to t_{n+1}.
7. $t_n = t_{n-1} + \Delta t_n$.

The assumption in (5.2) is that τ_s changes slowly with time. Otherwise, it should be evaluated simultaneously with τ_t and the step recomputed if τ_s or τ_t is too large. The iterations are teminated at iteration m when the residual in (4.3) is sufficiently small compared to ϵ. After a time step is complete according to algorithm (5.2), the error equation (2.14) is solved on the next coarser grid with estimates of τ_s and τ_t.

6 Numerical results

The algorithm in Section 5 is executed on 16 processors of a Sun Ultra Enterprise 6000 server. Two examples in two dimensions illustrate the behaviour of our method.

6.1 Rotating pulse

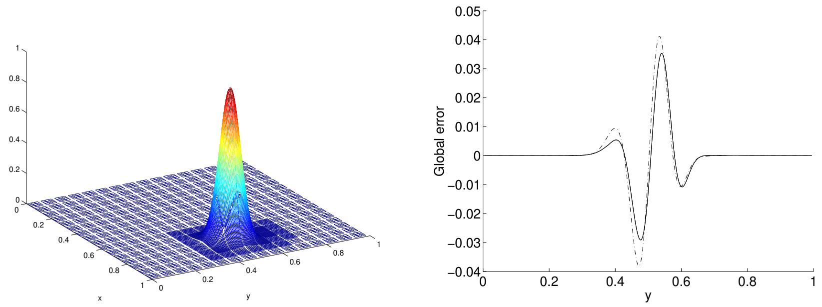

Figure 6.1: The Gaussian pulse after one period (left) and the exact (dash-dotted) and estimated (solid) global error in the solution at $x = 0.75$ (right).

The first example is the scalar equation

$$u_t + a_1 u_x + a_2 u_y = 0,\ (x, y) \in [-1, 1] \times [-1, 1]. \tag{6.1}$$

The velocity vector is $a = 2\pi(-\sin\psi, \cos\psi)^T$ where $\psi = \arctan y/x$. The initial solution u^0 at $t = 0$ is a Gaussian pulse. In the analytic solution, the pulse is rotated about the origin without distortion and $u(t+1) = u(t)$.

The grid is partitioned into 16×16 blocks. Each block has three grid levels with 12×12 to 48×48 cells. In Figure 6.1, the pulse has returned to its initial position and the exact error in the solution is compared to the estimated error obtained by integrating the error equation (2.14) on a coarser grid with estimates of the discretisation errors τ_s and τ_t as in (2.16) and (2.21). The local error is bounded by $\|\tau_s + \tau_t\| \le 2\epsilon$. Here, $\epsilon = 0.02$ and the local error is comparable to the global error at $t = 1$.

The midpoints of every third cell are combined in each block at $t = 0$ and $t = 1$ in Figure 6.2. The adapted grids are not identical but close to each other.

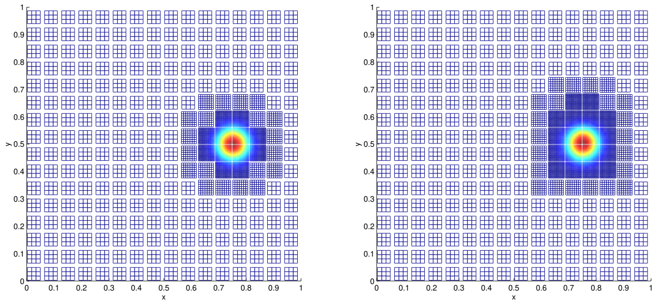

Figure 6.2: The adapted grid for the Gaussian pulse initially, $t = 0$, (left) and after one period, $t = 1$, (right).

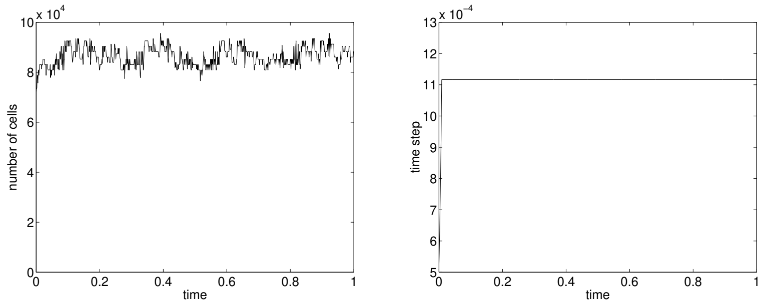

Figure 6.3: The number of cells in the adapted grid (left) and the time step (right) for the Gaussian pulse.

The total number of cells in the adapted grid in one period is displayed in Figure 6.3. It oscillates but is less than 10^5. The average number of cells including the coarse grid cells for the spatial error estimate is $1.19 \cdot 10^5$. A uniform grid with the fine step size in every block has almost $5.9 \cdot 10^5$ cells. The time step is constant except for the first step. The explanation is that the maximum time derivatives of the solution in the local error remain the same for the period.

The elapsed time was 1354 s for an adapted grid with three levels and a finest grid with 40×40 cells using 16 processors with dynamic load balancing. For comparison, the solution on a uniform grid with a resolution corresponding to 40×40 cells was computed with one block per processor and no error estimates to minimize the administration. The number of cells was $4.1 \cdot 10^5$ and it took 2362 s.

6.2 Linearised Euler equations

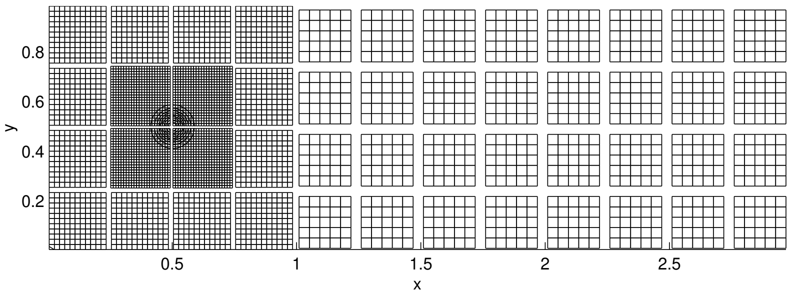

Figure 6.4: The adapted grid and isolines of u_3 at $t = 0$ in the channel.

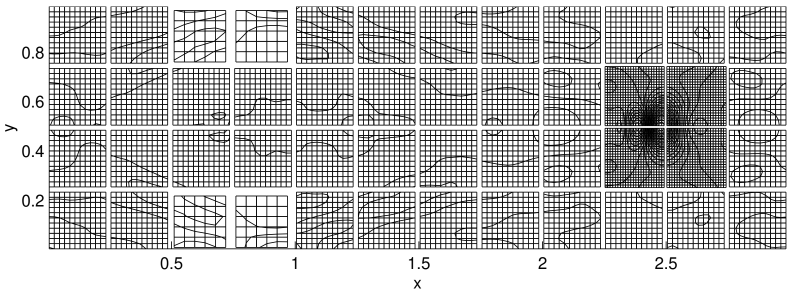

Figure 6.5: The adapted grid and isolines of u_3 at $t = 2$ in the channel.

In this example, the system of equations is

$$u_t + \begin{pmatrix} 1 & c & 0 \\ c & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} u_x + \begin{pmatrix} 0 & 0 & c \\ 0 & 0 & 0 \\ c & 0 & 0 \end{pmatrix} u_y = 0. \tag{6.2}$$

We choose $c = 2$, simulating the development of perturbations in subsonic flow (see [22]). Let $u = (u_1, u_2, u_3)^T, v_1 = u_1 + u_2$ and $v_2 = u_1 - u_2$. Then the boundary conditions are

$$\begin{aligned} v_1(0,y) &= 0,\ u_3(0,y) &= 0,\ v_2(3,y) &= 0, \\ u_3(x,0) &= 0,\ u_3(x,1) &= 0. \end{aligned}$$

For the remaining numerical boundary conditions we use linear extrapolation.

The initial distribution at $t = 0$ is a Gaussian pulse in u_3 and $u_1 = u_2 = 0$. The grid consists of 12×4 blocks with three grid levels with 48×48 cells in each block at the finest level. The grid and isolines of u_3 at $t = 0$ and $t = 2$ are found in Figures 6.4 and 6.5. Lines are drawn between every second cell midpoint in each block in the figure. The pulse is propagated downstream and the energy in the initial solution is redistributed from u_3 to u_1 and u_2.

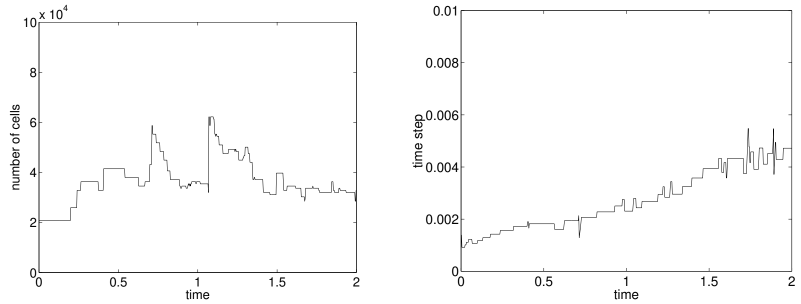

Figure 6.6: The number of cells in the adapted grid (left) and the time step (right) for the channel solution.

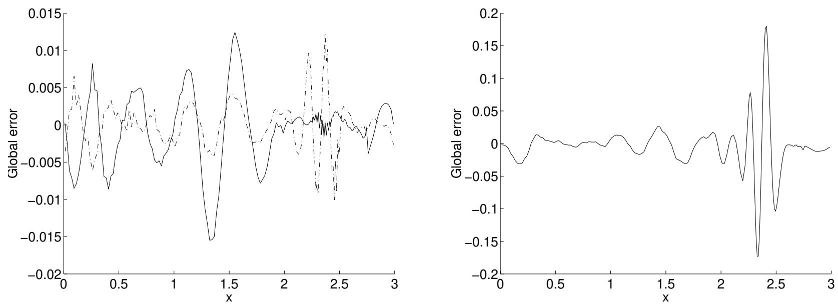

Figure 6.7: The estimated global error in u_1 (solid) and u_2 (dotted) (left) and in u_3 (right) in the middle of the channel at $t = 2$.

The number of cells and the time steps in the interval $[0, 2]$ are displayed in Figure 6.6. The adapted grid including the coarse grid for the error estimate has on average $0.45 \cdot 10^5$ cells. On 16 processors the elapsed time is 1337 s. The time step grows slowly due to the redistribution and a decay of higher derivatives. On a uniform grid with the finest resolution in every block, the number of cells would be $1.1 \cdot 10^5$. The elapsed time on this grid is 2271 s with a constant time step of 0.001 and the grid partitioned into 16 blocks.

The integrated global error in u_i is shown in Figure 6.7. The error tolerance ϵ is 0.1. The error is concentrated at the Gaussian pulse.

7 Conclusion

A second order numerical scheme for adaptive solution of hyperbolic PDEs has been derived. It adapts the step sizes and time steps so that a tolerance is satisfied by estimated discretisation errors. The grid is refined in blocks with jumps in the step size at the block boundaries. Special care is necessary there to ensure stability and second order accuracy. The implicit time stepping method generates a system of equations to be solved at each time step. The solution is computed iteratively by GMRES. The method is used to successfully solve two linear two-dimensional examples in parallel. In both examples, the algorithm selects a grid and a time step adapted to the solution of the equations with a global error estimate. A constant time step and a uniform static grid could achieve the same accuracy in the solution, but the computational cost is about twice as high, the number of cells is $2-3$ times higher, and only an experienced user can choose the appropriate constant uniform grid and time step.

REFERENCES

1. M. Berger, *Stability at interfaces with mesh refinement*, Math. Comput., 45 (1985), pp. 301–318.
2. M. Berger, *On conservation at grid interfaces*, SIAM J. Numer. Anal., 24 (1987), pp. 967–984.
3. M. Berger, P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comput. Phys., 82 (1989), pp. 64–84.
4. M. Berzins, *Global error estimation in the method of lines for parabolic equations*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 687–703.
5. K. E. Brenan, S. L. Campbell, L. R. Petzold, *Numerical solution of initial-value problems in differential-algebraic equations*, North-Holland, New York, 1989.
6. P. Colella, M. R. Dorr, D. D. Wake, *Numerical solution of plasma fluid equations using locally refined grids*, J. Comput. Phys., 152 (1999), pp. 550–583.
7. G. Dahlquist, *G-stability is equivalent to A-stability*, BIT, 18 (1978), pp. 384–401.
8. R. L. Davis, J. F. Dannenhoffer, III, *3-D adaptive grid embedding Euler technique*, AIAA-paper 93-0330, 1993.
9. J. De Keyser, K. Lust, D. Roose, *Run-time balancing support for a parallel multi-block Euler/Navier-Stokes code with adaptive refinement on distributed memory computers*, Parallel Comput., 20 (1994), pp. 1069–1088.
10. D. De Zeeuw, K. G. Powell, *An adaptively refined Cartesian mesh solver for the Euler equations*, J. Comput. Phys., 104 (1993), pp. 56–68.
11. K. Eriksson, D. Estep, P. Hansbo, C. Johnson, *Introduction to adaptive methods for differential equations*, Acta Numerica, (1995), pp. 105–158.
12. L. Ferm, P. Lötstedt, *Blockwise adaptive grids with multigrid acceleration for compressible flow*, AIAA J., 37 (1999), pp. 121–123.

13. L. Ferm, P. Lötstedt, *Adaptive error control for steady state solutions of inviscid flow*, Report 2000-003, Dept. of Information Technology, Uppsala University, Uppsala, Sweden, 2000.
14. B. Gustafsson, *The convergence rate for difference approximations to mixed initial boundary value problems*, Math. Comput., 29 (1975), pp. 396–406.
15. E. Hairer, S. P. Nørsett, G. Wanner, *Solving ordinary differential equations*, 2nd ed., Springer-Verlag, Berlin, 1993.
16. L. Hemmingsson, *A fast modified sine transform for solving block-tridiagonal systems with Toeplitz block*, Numer. Algorithms, 7 (1994), pp. 375–389.
17. L. Hemmingsson, *Toeplitz preconditioners with block structure for first-order PDEs*, Numer. Linear Algebra Appl., 3 (1996), pp. 21–44.
18. S. Holmgren, K. Otto, *A comparison of preconditioned iterative methods for nonsymmetric block-tridiagonal systems of equations*, Report No. 123 (revised), Department of Scientific Computing, Uppsala University, Uppsala, Sweden, 1989.
19. M. Holmström, *Solving hyperbolic PDEs using interpolating wavelets*, SIAM J. Sci. Comput., 21 (1999), pp. 405–420.
20. R. D. Hornung, J. A. Trangenstein, *Adaptive mesh refinement and multilevel iteration for flow in porous media*, J. Comput. Phys, 136 (1997), pp. 522–545.
21. G. Karypis, K. Schloegel, V. Kumar, *Parmetis–Parallel graph patitioning and sparse matrix ordering library*, Version 2.0, Univ. of Minnesota, Minneapolis, MN, 1998.
22. H.-O. Kreiss, J. Lorentz, *Initial-Boundary Value Problems and the Navier-Stokes Equations*, Academic Press, New York, 1989.
23. P. Lax, B. Wendroff, *Systems of conservation laws*, Comm. Pure Appl. Math., XIII (1960), pp. 217–237.
24. F. Liu, S. Ji, G. Liao, *An adaptive grid method and its application to steady Euler flow calculations*, SIAM J. Sci. Comput., 20 (1998), pp. 811–825.
25. P. Lötstedt, S. Söderberg, *Parallel solution of hyperbolic PDEs with space-time adaptivity*, in Finite Volumes for Complex Applications II, eds R Vilsmeier, F. Benkhaldoun, D. Hänel, Hermes, Paris, 1999, pp. 769–776.
26. S. Muzaferija, D. Gosman, *Finite-volume CFD procedure and adaptive error control strategy for grids of arbitrary topology*, J. Comput. Phys., 138 (1997), pp. 776–787.
27. J. Oliger, X. Zhu, *Stability and error estimation for component adaptive grid methods*, Appl. Numer. Math., 20 (1996), pp. 407–426.
28. E. Pärt-Enander, B. Sjögreen, *Conservative and non-conservative interpolation between overlapping grids for finite volume solutions of hyperbolic problems*, Computers & Fluids, 23 (1994), pp. 551–574.
29. K. G. Powell, *A tree-based adaptive scheme for solution of the equations of gas dynamics and magnetohydrodynamics*, Appl. Numer. Math., 14 (1994), pp. 327–352.
30. Y. Saad, M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
31. J. Steensland, S. Söderberg, M. Thuné, *A comparison of partitioning schemes for block-based parallel SAMR algorithms*, to appear in Proc. of Applied Parallel Computing (PARA2000), Bergen, 2000.
32. J. Wu, H. Ritzdorf, K. Oosterlee, B. Steckel, A. Schüller, *Adaptive parallel multigrid solution of 2D incompressible Navier-Stokes equations*, Int. J. Num. Meth. Fluids, 24 (1997), pp. 875–892.