A SIMPLE AND EFFICIENT SPACE-TIME ADAPTIVE GRID TECHNIQUE FOR UNSTEADY COMPRESSIBLE FLOWS

Jeroen Wackers
junior researcher
Delft University of Technology, Faculty of Aerospace Engineering
P.O. Box 5058, 2600 GB Delft, The Netherlands

Barry Koren
senior researcher
CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands, and
Delft University of Technology, Faculty of Aerospace Engineering
P.O. Box 5058, 2600 GB Delft, The Netherlands
senior member AIAA

ABSTRACT

A space-time adaptive gridding technique for unsteady flows is presented. The technique is applied to the 2D unsteady Euler equations. The method is relatively simple, computationally very efficient and it can be easily adapted to other types of fluid flow. It consists of four parts: (i) a time-stepping algorithm that adapts the grid to the solution several times per coarse time step, (ii) a second-order accurate discretisation of the flow equations that combines a limited upwind discretisation of the fluxes with a two-step discretisation of the time derivatives and is well-suited for adapted grids, (iii) a simple data structure to store the solution and the grid geometry, and (iv) a refinement criterion. Two of these are tested, one based on the first and one on the second spatial derivative of the density. Results for two test problems, the classical forward-facing step problem and the shedding of vorticity from a flat plate, show that the method is much more efficient than comparable methods without adaptive gridding.

NOMENCLATURE

SYMBOLS

$C_{\partial\rho}$	gradient ρ refinement criterion
$C_{\partial^2\rho}$	second derivative ρ refinement criterion
C_n	normal force coefficient, $\frac{\text{normal force}}{\frac{1}{2}\rho U^2 \bar{c}}$
$\boldsymbol{f}, \boldsymbol{g}$	flux vector, horizontal / vertical
$\boldsymbol{F}_O, \boldsymbol{G}_O$	Osher's approximate flux vector
l	level of refinement
M	maximum level of refinement
p	pressure
$\boldsymbol{q}$	state vector
t	time
u, v	velocity components in x- and y-direction
x, y	spatial coordinates
z	unscaled entropy
γ	ratio of specific heats
Δt	time step
$\Delta x, \Delta y$	cell sizes, x- and y-direction
ρ	density
Ω	spatial domain, cell or collection of cells
$\partial\Omega$	boundary of Ω

SUBSCRIPTS

b, r, a, l	below, right, above, left (value of n)
d	diagonal
i	indicates a cell
n	neighbour
p	component of state vector

SUPERSCRIPTS

l	level of refinement
V	virtual cell, virtual state

INTRODUCTION

Adaptive grid refinement is a technique to speed up the numerical solution of partial differential equations by starting the solution on a coarse, uniform grid and refining this grid locally to accurately resolve areas with e.g. large gradients. All other areas are calculated on coarser grids, so computation time is saved. Adaptive gridding has been widely used already for the solution of fluid-flow problems, both steady and unsteady. For unsteady flow, pioneering work has been done by Berger et al.[1,2]. Many others have followed.

Here, a space-time adaptive grid technique for unsteady flows is presented, which distinguishes itself from existing techniques by the use of a very simple data structure, a likewise simple discretisation of the fluid-flow equations and a time-stepping algorithm that allows a very flexible and fast adaptation of the grid to the solution. The method is applied – to start with – to the 2D unsteady Euler equations of gas dynamics. The discretisation of the flow equations is largely taken from earlier work at CWI[4,6,9] and adapted here for unsteady flow. The data structure and the algorithm are new developments. A full overview of the

method is given in[12].

This paper describes the different parts of the method separately. After an overview of the flow equations used, the philosophy of adaptive gridding and the algorithm that handles the adapted grid are described in the third section. The next section describes the discretisation of the flow equations, it focuses on the discretisation of the time derivatives and the extra features needed to make the discretisation suitable for adapted grids. The following two sections describe the data structure and give two refinement criteria, which are used to determine where the grid is refined. Results in the last section are used to prove that the current method is indeed more efficient than a comparable method without adaptive gridding.

FLOW EQUATIONS

The governing flow equations used in this paper are the 2D unsteady Euler equations of gas dynamics:

$$\frac{\partial}{\partial t}\boldsymbol{q} + \frac{\partial}{\partial x}\boldsymbol{f}(\boldsymbol{q}) + \frac{\partial}{\partial y}\boldsymbol{g}(\boldsymbol{q}) = 0, \qquad (1)$$

with

$$\boldsymbol{q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}, \quad \boldsymbol{f} = \begin{pmatrix} \rho u \\ p + \rho u^2 \\ \rho uv \\ \rho uH \end{pmatrix}, \quad \boldsymbol{g} = \begin{pmatrix} \rho v \\ \rho uv \\ p + \rho v^2 \\ \rho vH \end{pmatrix}. \qquad (2)$$

The specific internal energy E and enthalpy H are

$$E = \frac{1}{\gamma - 1}\frac{p}{\rho} + \frac{1}{2}\left(u^2 + v^2\right), \quad H = E + \frac{p}{\rho}. \qquad (3)$$

Integrating equation (1) over a rectangular domain gives

$$\frac{\partial}{\partial t}\iint_{\Omega} \boldsymbol{q}\, dx\, dy - \int_{\partial\Omega_b} \boldsymbol{g}\, dx + \int_{\partial\Omega_r} \boldsymbol{f}\, dy + \int_{\partial\Omega_a} \boldsymbol{g}\, dx - \int_{\partial\Omega_l} \boldsymbol{f}\, dy = 0, \qquad (4)$$

which says that the time rate of change of the state $\boldsymbol{q}$ in a rectangular domain is equal to the net inflow over its four boundaries below, right, above and left.

Equation (4) is discretised on a rectangular grid. A domain Ω is divided into rectangular cells Ω_i, an average state $\boldsymbol{q}_i$ is defined in each cell and approximations to the fluxes in the cell faces are calculated using Osher's approximate Riemann solver[4,8]. Then equation (4) is used to advance the states in the cells one time step.

This discretisation must be conservative. This means that the flux in a cell face out of one cell must be equal to the flux into the next cell. But, on an adaptive grid, this has more implications, as we will see later on.

ADAPTIVE GRIDDING

BACKGROUND

Most solutions of the Euler equations have a few areas where the flow changes rapidly, like shock waves. To compute an accurate solution in these areas, a very fine grid is needed. On the other hand, the areas without rapid changes can be solved with the same accuracy on a much coarser grid.

The idea behind adaptive gridding is to compute only the areas with rapid changes on a fine grid and the rest of the solution on a coarser grid, thus reducing the total computation time. Here, adaptation through enrichment is used: the calculation is started on a very coarse basic grid and this grid is refined by dividing the coarse cells into four smaller cells when this is required. For stability reasons, the time step for these smaller cells is twice smaller too, so the grid is adapted in space and time (figure 1). The smaller cells can be split again to form even smaller cells and, when the fine grid is not needed anymore, they are merged into one big cell again.

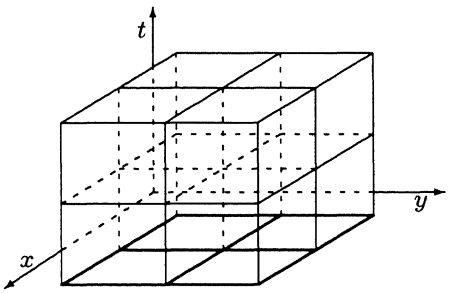

Figure 1: A big cell is split into four smaller cells with a smaller time step.

Figure 1 shows that it takes eight times more computational work to advance the refined cells a single coarse time step than the basic coarse cell. And for cells that are n times refined, this is 2^{3n} times more computational effort. So when cells are refined, the total computation time increases rapidly. Therefore, the grid is adapted to the solution several times per coarse time step, instead of once[11] or less than once[1]. If the grid can change rapidly, wave patterns can be followed with very narrow strips of refined cells, thus keeping the total number of cells low.

A TIME STEP

The current algorithm is illustrated here with a simple, 1D example. Consider the grid of figure 2a: one unrefined cell, a cell that is refined once and two cells that are refined twice. We will say that these cells are on 'level' 0, 1 and 2, respectively. All fluxes across the cell faces are calculated (b). The only cells that can be advanced in time now are the smallest cells (c), because the fluxes into the larger cells are not yet known for their entire time steps. Then the fluxes into the smallest cells are calculated again with the

new states in the small cells (d) and the smallest cells are advanced again (e). But now, the flux out of the level 1 cell is known: conservation requires that it is the summation of the fluxes into its smaller neighbour cell, over that cell's two time steps. So the level 1 cell is advanced too (f). This whole procedure is repeated (g) and now, the fluxes into the level 1 cell can be summed to give the flux into the level 0 cell, to advance this cell. The coarse time step is now complete.

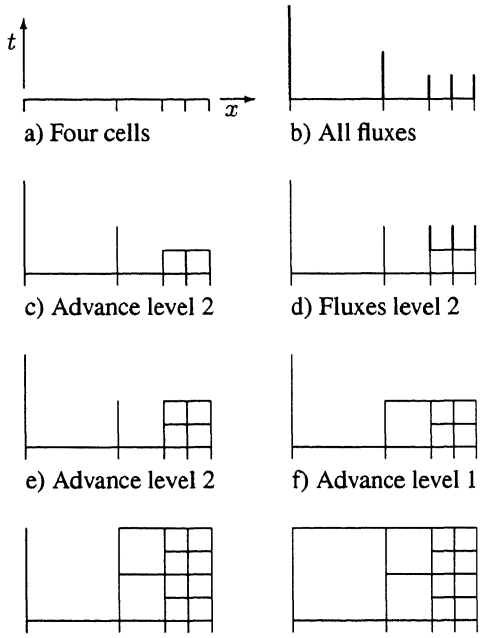

Figure 2: Advancing the cell states, one coarse time step.

In this example, the grid does not change in time. Let us consider where the cells can be refined (figure 3). Before the time step starts, the level 0 cell can be split into level 1 cells (a) or the level 1 cell can be split into level 2 cells (b). (In the algorithm, this last refinement will not be allowed, because it gives a level jump of two between neighbour cells, which causes large errors.) The level 1 cell can even be refined at another time: after it has been advanced once (c). Summarizing: a cell can be split after each of its *own* time steps. The cells on the highest level allowed (here level 2) cannot be split. Also, example (c) shows that the grid can be adapted more than once per coarse time step.

And when may cells be unrefined (figure 4)? Before the time step starts, the level 2 cells can be merged into one level 1 cell (a), but the level 1 cell cannot be merged. It must be merged with another level 1 cell, but there are two level 2 cells where that cell should be. After one of their time steps, the level 2 cells cannot be merged (b), because the resulting level 1 cell does not fit in the level 1 time steps, but after two time steps, the level 2 cells can be merged (c). Summarizing: cells can also be merged after each of their own time steps. Only, the cells on the lowest level that was just advanced (level 2 in (b), level 1 in (c)) cannot be merged. And all cells to be merged should *not* be split again.

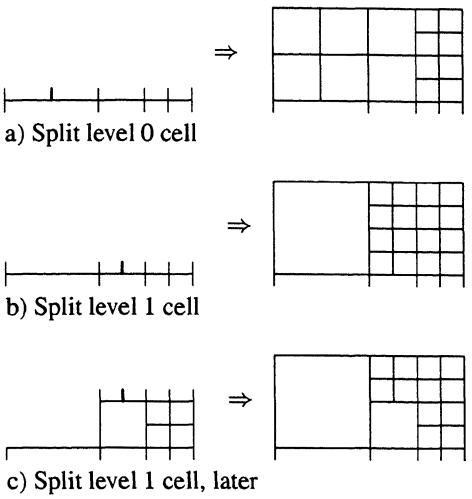

Figure 3: Refinement possibilities.

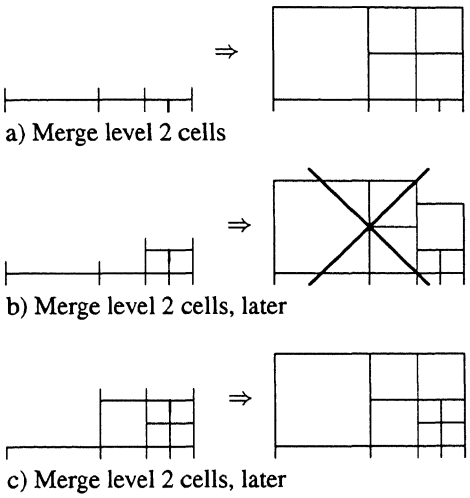

Figure 4: Unrefinement possibilities.

ALGORITHM

These principles are used in an algorithm, that is given here in pseudo-code. The operations are performed on the group of all cells that is currently on a certain level. The reader is encouraged to follow the previous example with this algorithm.

```
Program Sagseo
do k = 1 to k_max call Timestep(M)
end Sagseo

Subroutine Timestep(l_now)
advance level l_now
if l_now advanced for first time then
    do i = l_now to M - 1 refine_check level i
    do i = l_now + 1 to M unrefine_check level i
    do i = l_now to M fluxes level i
    if l_now = 0 return, time step finished
    call Timestep(M)
```

else (second time)
  call Timestep(l_now - 1)
end if
end Timestep

SECOND-ORDER DISCRETISATION

A second-order accurate discretisation of the flow equations is used, an approximate equation in which the average states in the cells are the only unknowns, which has a difference of $\mathcal{O}(h^2)$ with equation (4). The fluxes are discretised with Osher's approximate Riemann solver[4,8], combined with a second-order limited approximation to the states at the cell faces[10]. This procedure is well-known. This section concentrates on the aspects of the flow equation discretisation that are special for adaptive gridding: the time derivative discretisation, the fluxes between large and small cells and the state in newly refined or unrefined cells.

TIME DISCRETISATION

For the time-derivative discretisation, a new scheme is used, based on the two-step Richtmyer scheme, but combined with the limited fluxes:

$$\begin{aligned} q_i^{k+1} &= q_i^k - \left(F_{O\,r}^k - F_{O\,l}^k\right)\frac{\Delta t}{\Delta x} \\ &\qquad - \left(G_{O\,a}^k - G_{O\,b}^k\right)\frac{\Delta t}{\Delta y}, \\ q_i^{k+2} &= q_i^k - 2\left(F_{O\,r}^{k+1} - F_{O\,l}^{k+1}\right)\frac{\Delta t}{\Delta x} \\ &\qquad - 2\left(G_{O\,a}^{k+1} - G_{O\,b}^{k+1}\right)\frac{\Delta t}{\Delta y}. \end{aligned} \tag{5}$$

The first step is first-order accurate, but the resulting state is only used to calculate the fluxes for the second step, which is second-order accurate.

This scheme is better than the related leapfrog scheme: it is more stable and it is self-starting (no separate equation is needed for the *first* time step). The latter is a great advantage on an adaptive grid, because a 'startup' situation occurs every time after a cell is split into four smaller cells. It is also better than the various Runge-Kutta type multistage schemes, which are hard to implement on a grid with cells that have different time steps.

Due to its two-step structure, the scheme can be easily combined with the algorithm of the previous section, that also has a two-step structure. In the algorithm, the first part of equation (5) is used to advance cells for the first time and the last part of equation (5) is used when the cells are advanced for the second time.

For a simplified equation, the stability of the scheme is proven for CFL numbers $\lambda_{max}\frac{\Delta t}{\Delta x} \leq 0.25$.

VIRTUAL STATES

For the second-order accurate calculation of the fluxes, the limited scheme uses an interpolation between several cells to approximate the state at a cell face. On a uniform grid, this is relatively straightforward, but problems arise on an adapted grid, when two neighbour cells have different sizes (figure 5). To calculate the flux across the face of the smaller cell, we need the states in the cells that are drawn with dashed lines. But these cells do not exist. Therefore, the only way to find these states is by interpolation. For steady problems, this interpolation has already been used by Van der Maarel[7]. It is extended here to be suitable for unsteady problems.

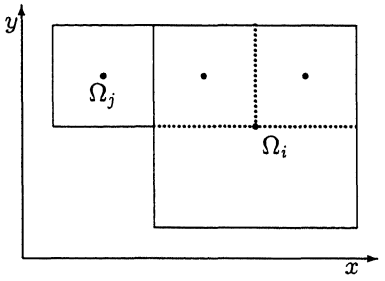

Figure 5: Big cell with smaller neighbour and two virtual cells.

The state in the virtual cell is interpolated between the big cell Ω_i and a diagonal cell Ω_d (figure 6). For second-order accuracy, a linear interpolation is used

$$q_i^V = q_i + (q_d - q_i)\frac{x_i^V - x_i}{x_d - x_i}, \tag{6}$$

with x a coordinate of the cell centres. In this equation, it does not matter if the diagonal cell is smaller, equally large, or larger than the cell Ω_i. Using the level l of the cells, equation (6) is rewritten as

$$q_i^V = q_i + \frac{1}{2 + 2^{l_i - l_d + 1}}\left(q_d - q_i\right). \tag{7}$$

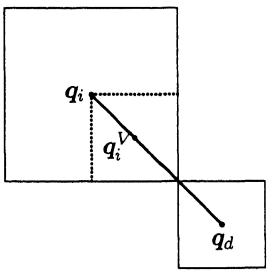

Figure 6: Interpolation of a virtual state.

This virtual state is only correct when the state in the big cell is known at the same time as the state in the smaller neighbour cell Ω_j. But the smaller cell has smaller time steps, so this is not true when the smaller cell is advanced once (figure 7). In this case, the virtual state is found by time-stepping the virtual state from equation (7). But it is easier, and still second-order accurate, to make *half* a time

step for the big cell and to add this to the virtual state:

$$q_i^V = q_i + \frac{1}{2 + 2^{l_d - l_i + 1}} (q_d - q_i) - \frac{1}{2} (F_{O\,i,r} - F_{O\,i,l}) \frac{\Delta t}{\Delta x} - \frac{1}{2} (G_{O\,i,a} - G_{O\,i,b}) \frac{\Delta t}{\Delta y}. \quad (8)$$

Note that the first-order accurate Euler discretisation is used for Δq_i^V, but for a *single* Δq of $\mathcal{O}(\Delta t)$ this discretisation gives an error of $\mathcal{O}(\Delta t^2)$, which is small enough.

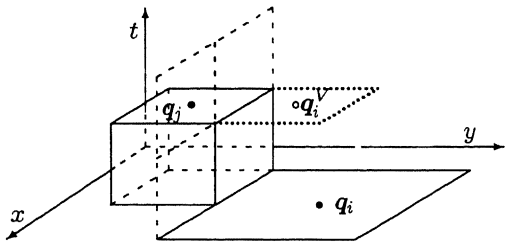

Figure 7: Virtual state that requires a time step.

INITIALISING STATES

When a cell is split into four, or when four cells are merged into one, then states must be chosen in these new cells and these new states must be second order accurate. But they must also be conservative: no mass etc. may disappear when the cells are split or merged.

When a cell is split into four smaller cells, then it is a logical choice to use the virtual states from the previous section to set the states in the small cells. But these virtual states are not necessarily conservative, because all four states are set with different interpolations. This is solved by setting the states in two opposite diagonal cells from the *same* linear interpolation: first, the virtual state from equation (7) is calculated for both cells and then, for each component q^p of $\boldsymbol{q}$, the cell is selected in which the virtual state has the smallest absolute difference from q_i^p. That state is kept and the state in the other cell is set as

$$q^p_{\text{other cell}} = 2q_i^p - q^p_{\text{one cell}}. \quad (9)$$

When this is done for both diagonals, then the average of the four new states is equal to $\boldsymbol{q}_i$, so the interpolation is conservative.

When four cells are merged into one cell, then the state in this new big cell is set by simply averaging the four old states.

GRID DATA STRUCTURE

To store an adapted grid and a solution on such a grid, a special data structure is needed. Adapted grids are irregular, therefore they cannot be stored in normal i,j-arrays. And for unsteady problems, the grid changes in time, so the storage space has to change in time too.

Here, the cell data (like the cell state, the fluxes and geometrical information) are stored in 1D arrays. In each array, the same position corresponds to the same cell, but the cells are distributed in the arrays at random. The arrays are bigger than the total number of cells and a list is kept of all the array locations that are 'free'. When a cell is split, three locations are taken from this 'free' list and used for three of the four new cells. The memory location of the original coarse cell is used for the fourth new cell. When four cells are merged again, then the three locations that are not used anymore are added to the 'free' list, ready to be used again.

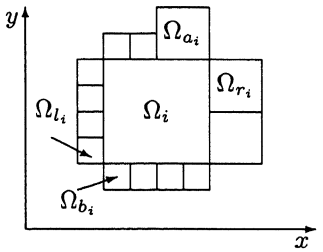

Figure 8: Four neighbour pointers for a cell with smaller neighbours.

The grid geometry is defined by six integer arrays per cell, that give a cell's relation to other cells. These are:

- the cell level.
- four neighbour pointers, one for each cell face. If the neighbour cells on a cell face are smaller, then we store (figure 8)
  - below: the leftmost cell,
  - right: the highest cell,
  - above: the rightmost cell,
  - left: the lowest cell.

  With this arrangement, it is easy to find the other cells on that face. Consider, e.g., the right face. The highest cell on this face has the neighbour pointer. The neighbour *below* of this neighbour cell is the *leftmost* cell below it, so it also lies next to the big cell. And this works for all other faces too. The neighbour pointers, together with the cell levels, are enough to reconstruct the entire grid geometry.
- one mother pointer to the cell from which the cell was split. When a cell is split into four, then the cell below left keeps the mother pointer from the big cell, the other three cells get a mother pointer to this cell (figure 9). The mother pointers are used when cells are merged, to make sure that the four cells to be merged were once split from the same cell. We find these cells as follows:
  1. Find a cell whose left neighbour is equal to its mother. Then we know that this cell was the cell below right when it was split *and* that the mother cell is not split again.

2. From these cells, select those for which the cell, the left neighbour, the neighbour above and the left neighbour of the neighbour above have the same level. Then we know that none of these four cells is split again, so they can be merged.

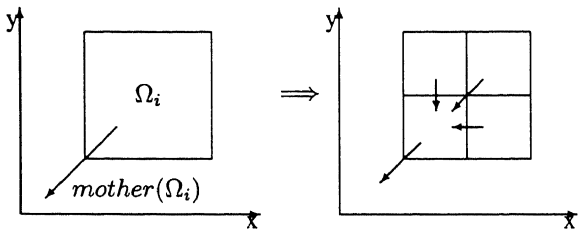

Figure 9: Mother pointers of a big cell and four refined cells.

REFINEMENT CRITERIA

The goal of an adaptive gridding procedure is to refine the grid locally, based on the solution. So far, we have seen how such an adapted grid is handled and how a solution is calculated on an adapted grid. But we also need some way to determine where the grid should be refined, based on the solution. This is done with a refinement criterion.

The ultimate goal of a refinement criterion is to provide the user with the best possible solution at the lowest possible computational costs. But the properties of a 'best solution' depend on what the user wants to see, like flow phenomena or force and moment data. Therefore, the choice of the refinement criterion must be made by the user.

Two different refinement criteria have been studied. They are described here, their behaviour is given in the next section.

GRADIENT ρ CRITERION

A simple, straightforward choice for a refinement criterion is to refine the grid in the neighbourhood of strong gradients in the solution. Here, the gradient of the density ρ, the first component of the state vector, is used, because the density changes in almost every type of flow pattern. Furthermore, it is the only state component that is the same for a stationary wave and a running wave with the same strength.

If the grid size is taken inversely proportional to the gradient of ρ, then $\frac{\partial\rho/\partial x}{\Delta x}$ and $\frac{\partial\rho/\partial y}{\Delta y}$ should be more or less constant in the entire computational domain. So we define a refinement criterion as the maximum gradient in a cell, estimated from the cell and its four neighbours, divided by the cell size,

$$C_{\partial\rho} = \max_{n=b,r,a,l} \left(\frac{|\rho_n - \rho_i|}{\frac{1}{2} + 2^{l_i - l_n - 1}} \right). \qquad (10)$$

This is, in fact, the undivided difference of ρ between the cells, corrected for a possible smaller or larger neighbour cell.

A maximum and minimum value are defined for $C_{\partial\rho}$. When $C_{\partial\rho}$ in a cell becomes higher than the maximum, the cell is refined. And if $C_{\partial\rho}$ drops below the minimum in four cells, then these are unrefined.

SECOND DERIVATIVE ρ CRITERION

It is an interesting idea to base a refinement criterion on an estimate of the local truncation error. For first-order discretisations of linear equations, this error is proportional to the second spatial derivative of the solution. And as the local solution error for a limited flux scheme has the shape of a first-order error (because the limiter adds numerical viscosity), the second derivative of the solution is a good choice for a refinement criterion. For the same reasons as before, ρ is used.

The second derivative criterion is defined as the maximum of two directions, horizontal and vertical,

$$C_{\partial^2\rho} = \max_{n=b,r} \left(\frac{\frac{\rho_n - \rho_i}{\frac{1}{2} + 2^{l_i - l_n - 1}} + \frac{\rho_i - \rho_{(n+2)}}{\frac{1}{2} + 2^{l_i - l_{(n+2)} - 1}}}{2^{-l_i} \Delta x \left(1 + 2^{l_i - l_n - 1} + 2^{l_i - l_{(n+2)} - 1}\right)} \right), \qquad (11)$$

where $n+2$ denotes the opposite neighbour of n. This is the well-known central discretisation of the second derivative, divided by the cell size. It is corrected to allow cells of different sizes. A maximum and minimum are defined as for the $C_{\partial\rho}$ criterion.

TWO CELLS PER LEVEL

It is undesirable to have neighbour cells with a level difference of two or more, as this may cause large errors and programming difficulties. To keep the grid smooth, some extra cells have to be refined sometimes where the refinement criterion is not yet too high. More precisely, extra cells are refined such that each band of cells at the same level is at least two cells wide:

Assume that a level difference of two is not allowed. Then, when a cell has a larger and a smaller neighbour and when it has just had the first of two time steps, that cell cannot be refined: if it was refined, then the larger cell has to be refined too and this is impossible, because it has only had half a time step. But when the smaller neighbour is refined, then the cell *must* be refined. This situation may never occur, so a cell may never have a larger *and* a smaller neighbour. When all groups of cells at the same level are at least two cells wide, this does not happen.

RESULTS

In this section, results are given for two test problems. One is the forward-facing step problem, known from the work of Woodward and Colella[13]. The other is a new problem, the shedding of vortices from a suddenly started flat plate. With these two problems, the features of the algorithm and the refinement criteria are illustrated. Furthermore,

they prove that the current method is more efficient than a method on a uniform grid.

FORWARD-FACING STEP

This first problem is used to validate the method and to compare the performance of the two refinement criteria. Therefore, the solutions obtained with the adaptive gridding algorithm are compared with solutions on a uniform grid that has the same grid size as the *smallest* cells in the adapted grid. These solutions are made with the same method as the adapted solutions, but with the grid adaptation turned off, so everything except the grid adaptation procedure is the same.

The forward-facing step problem was introduced by Emery[3] and later used, among others, by Woodward and Colella[13]. The problem starts with a Mach 3 flow in a wind tunnel section. At $t = 0$, a forward-facing step materializes in the floor of the section. A bow shock develops in front of this section. Later on, this shock reflects from the top surface.

The solution is studied at $t = 4$ (figure 10). At this moment, the shock has reflected three times and the first reflection has developed into a Mach stem with a normal shock and a trailing contact discontinuity. An expansion fan above the step interacts with the shocks. This fan is slightly overexpanded, so it ends in a weak shock. This oblique shock crosses the first reflected shock and merges with the second, causing a very weak trailing contact discontinuity.

A plot of the entropy (figure 10) reveals some aspects of the solution that are not visible in the density plot. Of course, the shocks show up, but especially the contact discontinuity is well visible. In this contact discontinuity, we see a slightly wavy pattern. These waves, known as Kelvin-Helmholtz instability, are a physical feature, although they are triggered by small numerical oscillations. On the surface above the step, a numerical boundary layer is visible, despite a correction that is applied above the step: in the first few cells behind the corner, the entropy and enthalpy are reset to the values in the last cell before the corner[12, 13]. Because of this correction, the boundary layer does not influence the shock reflection much.

The density solution with adaptive gridding is given in figure 11, both for the $C_{\partial\rho}$ and the $C_{\partial^2\rho}$ criterion. If we compare these with the uniform solution of figure 10, we see that they are comparable in accuracy. The shocks are sharp and well visible everywhere, as is the contact discontinuity. From the position of the shocks, it is seen that the $C_{\partial^2\rho}$ criterion performs slightly better. The normal shock is longer and placed a little more forward, which corresponds better to the uniform-grid solution.

In the entropy solutions (figure 12), differences in the numerical boundary layer can be seen. For the $C_{\partial\rho}$ criterion, the strength of the boundary layer is actually reduced before the shock reflection, so this solution is even better here than the uniform-grid solution. Further back, there is an erroneous 'curl' in the entropy solution. The Kelvin-Helmholtz instability is visible in both solutions, but it is damped and smeared more for the $C_{\partial^2\rho}$ criterion.

The adapted grids at $t = 4$ reveal much about the nature of the refinement criteria. The shocks are fully refined for both criteria, but the refinements are broader for the $C_{\partial^2\rho}$ criterion. This happens because the first derivative of ρ is high in the middle of a shock, while the second derivative is higher near the sides.

The expansion is refined more for the $C_{\partial\rho}$ criterion, because an expansion usually has a large gradient, but not much curvature. The last part of the contact discontinuity is not refined for the $C_{\partial^2\rho}$ criterion, which explains why the contact discontinuity is smeared there. None of the two criteria refines the second, weak contact discontinuity.

The expansion fan shows a weakness of the $C_{\partial^2\rho}$ criterion: its high sensitivity to small disturbances in the solution. Especially the refinement and unrefinement of cells causes errors. These errors are very small, but they cause peaks in the second derivative estimate, so the grid is refined more (or again). This 'self-induction' causes a ragged grid, which can be seen in the expansion fan.

CPU times for these solutions are measured on a SUN E250 workstation, see table 1. These times include all computations but no input or output. The uniform-grid computation has some unnecessary overhead for the needlessly complex data structure, but no refinement or unrefinement checks are done*.

	CPU time (s)	% of unif.
uniform grid	43,184	
$C_{\partial\rho}$ criterion	8554	20
$C_{\partial^2\rho}$ criterion	8981	21

Table 1: Forward-facing step, CPU times for the solution on a uniform grid and refined grids, with the $C_{\partial\rho}$ and the $C_{\partial^2\rho}$ refinement criterion.

The table shows that, for these settings of the upper and lower limit, both criteria require about the same computation time. The results above confirm that both solutions are comparable in accuracy to the uniform-grid solution, although the $C_{\partial^2\rho}$ solution is slightly better. So, in this case, the adaptive gridding method delivers this comparable accuracy at five times lower computational costs.

FLAT PLATE

As a second test problem, the subsonic flow around a flat plate is calculated. No results are known in advance from the literature, so in this sense, the problem resembles practice. The key question here is, whether the solver works under these conditions. Furthermore, it is a problem from an interesting new flow regime, with vortices, rather than

*It is estimated that the uniform-grid computation time can be reduced to about 80% of its current value if the code is rewritten for uniform grids.

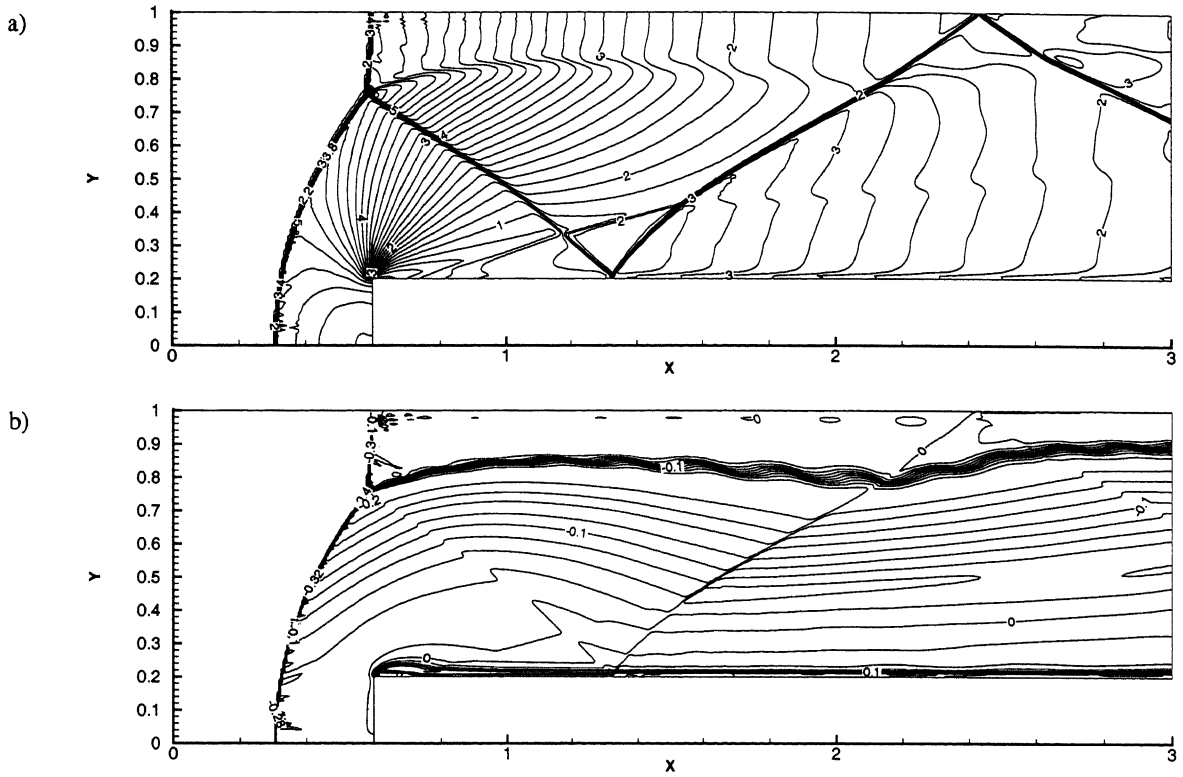

Figure 10: Forward-facing step, solution at $t = 4$ computed on a uniform grid with $\Delta x = 1/160$ and $\Delta x/\Delta t = 12.5$. Iso-lines for the density ρ (a) and unscaled entropy $z = \ln\left(\frac{p}{\rho^\gamma}\right)$ (b).

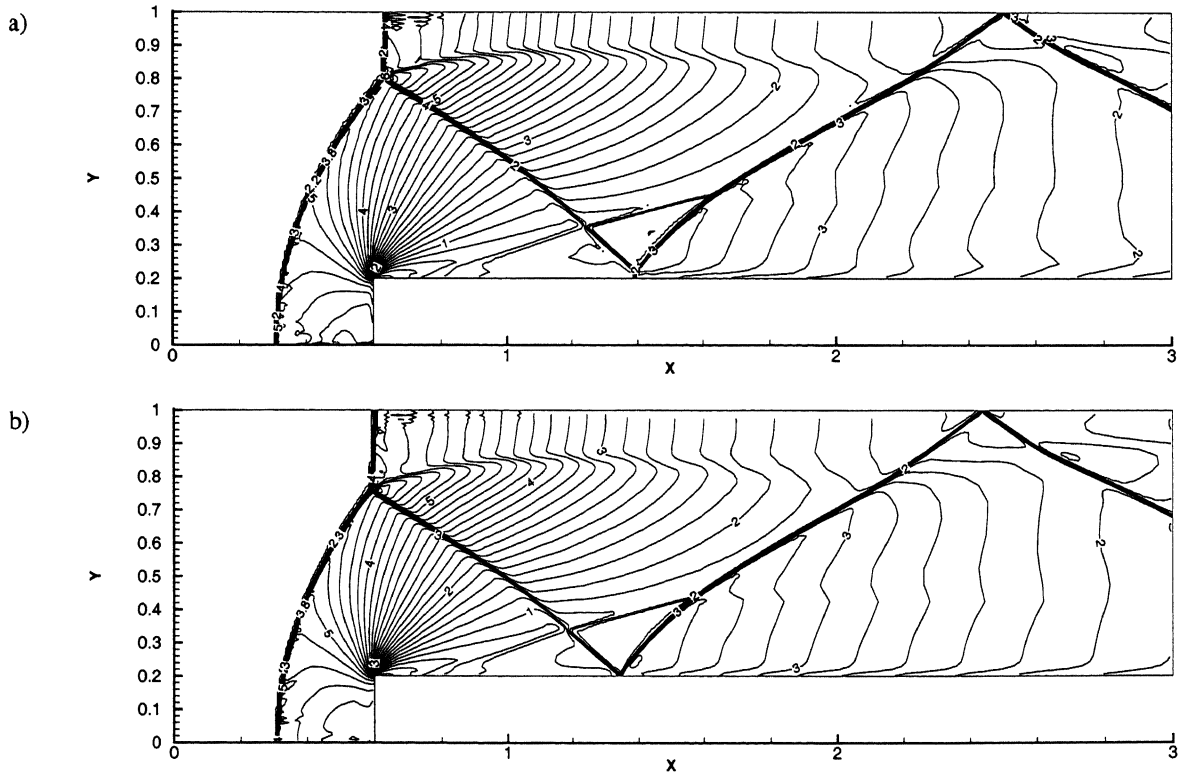

Figure 11: Forward-facing step, the density ρ at $t = 4$, computed on an adapted grid with the $C_{\partial\rho}$ criterion (a) and the $C_{\partial^2\rho}$ criterion (b). The basic coarse grid has $\Delta x = 1/20$ and $\Delta x/\Delta t = 12.5$. Maximum level of refinement is 3.

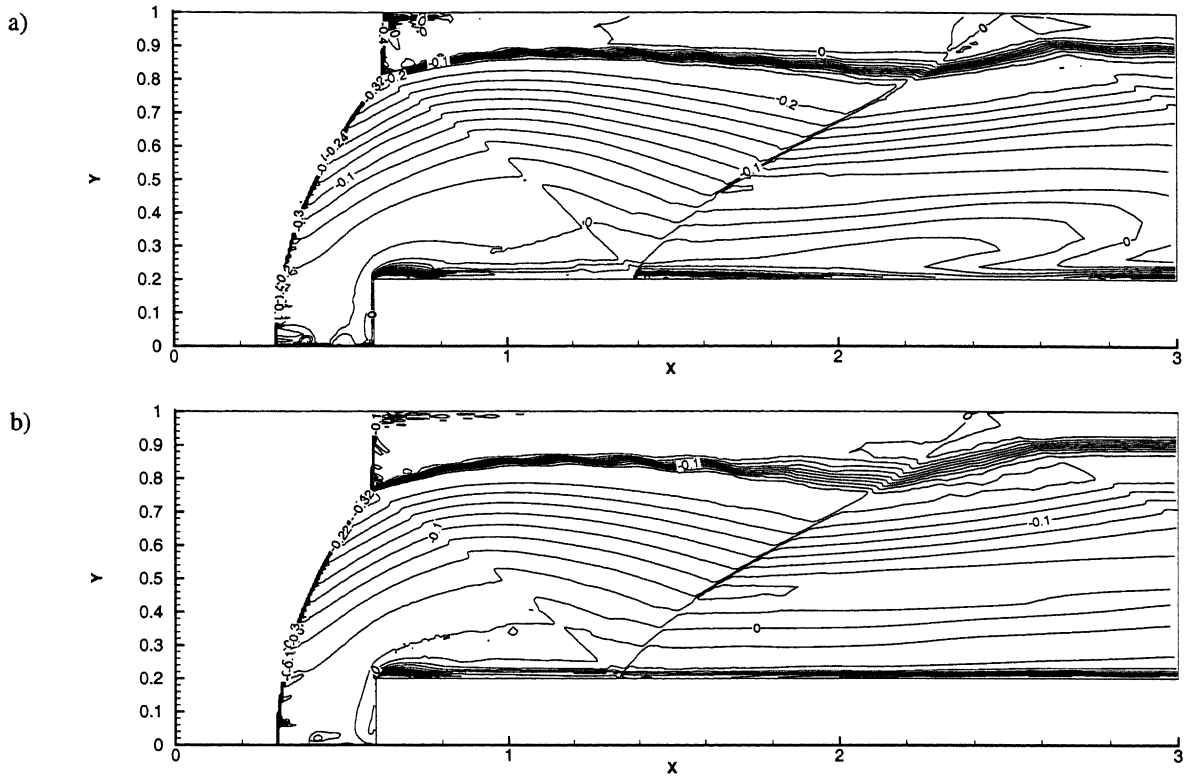

Figure 12: Forward-facing step, the unscaled entropy z at $t = 4$, computed on adapted grids with the $C_{\partial\rho}$ criterion (a) and the $C_{\partial^2\rho}$ criterion (b).

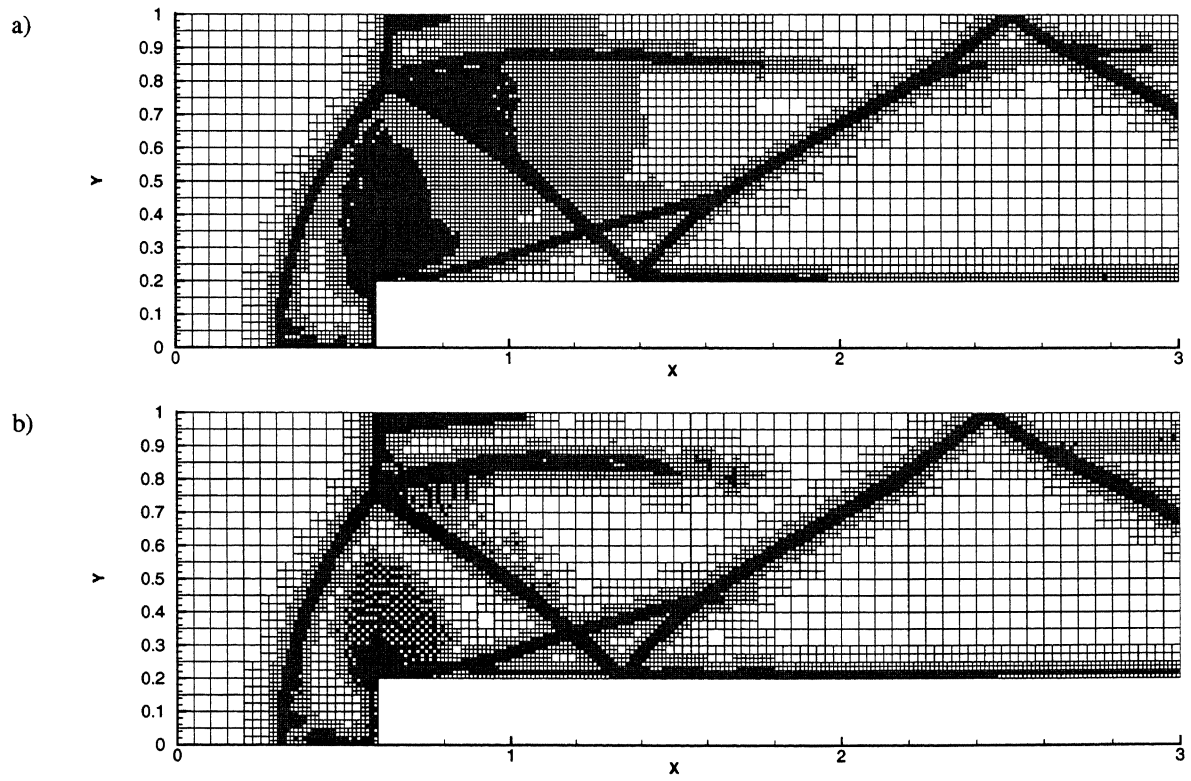

Figure 13: Forward-facing step, the adapted grids at $t = 4$ for the solution in figures 11 and 12. $C_{\partial\rho}$ refinement criterion (a) and $C_{\partial^2\rho}$ refinement criterion (b).

shocks, expansion waves and contact discontinuities. This section describes the method used to solve the flat plate problem.

PROBLEM DEFINITION A flat plate is placed at rest in a flow at rest, under an angle of attack α. At $t = 0$, the plate is instantaneously accelerated to Mach 0.5 and after $t = 0$, the speed is kept constant. This problem is modeled on a rectangular grid with a cut in it to represent the flat plate. Initially, a flow with an angle of attack α is specified in the whole domain.

CONVERGENCE STUDY Because the problem is new, a convergence study is done on uniform grids, on a small domain (7×6 chord lengths) and on a short time interval ($t = 3$). Three grids are used, with $\Delta x = \Delta y = 1/16$, $1/32$ and $1/64$ chord length. These tests show that, after some time, the flow separates from the leading edge. But until this separation, the flow is independent of the grid size and is converged for $\Delta x = 1/64$. For a low angle of attack, $\alpha = 5^o$, separation occurs quite late, so sensible results can be obtained after $t = 3$.

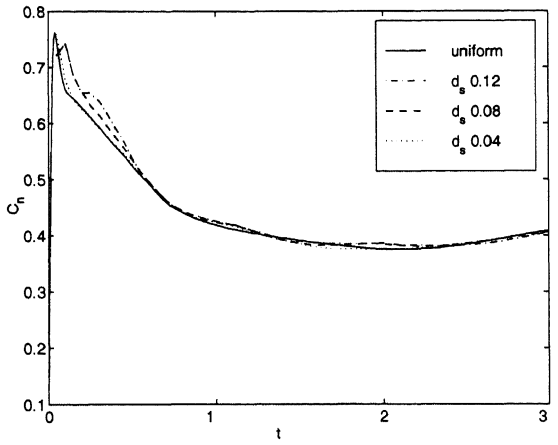

Figure 14: Flat plate, development of the normal force coefficient C_n for three settings of the upper and lower limit for the refinement criterion. The upper limit d_s is shown, the lower limit is always $0.4d_s$.

REFINEMENT CRITERION A gradient criterion is the most robust and stable choice for a refinement criterion, but $C_{\partial\rho}$ is not suitable because the density does not change much in this subsonic flow. Therefore, a gradient criterion is based on the second component of q, the momentum term ρu. A test is done with this criterion, on the same small domain as the convergence test. The upper and lower limit for the criterion are set by comparing the solution with different limits with the uniform solution.

The development of the normal force coefficient C_n on the plate in time is given in figure 14 for the case studied ($\alpha = 5^o$, maximum level of refinement 5 on a $\Delta x = 1/2$ basic grid, which corresponds to $\Delta x = 1/64$ for the smallest cells). The lowest value for the limit gives a solution that matches the uniform solution well, especially in the beginning. Therefore, this value is selected.

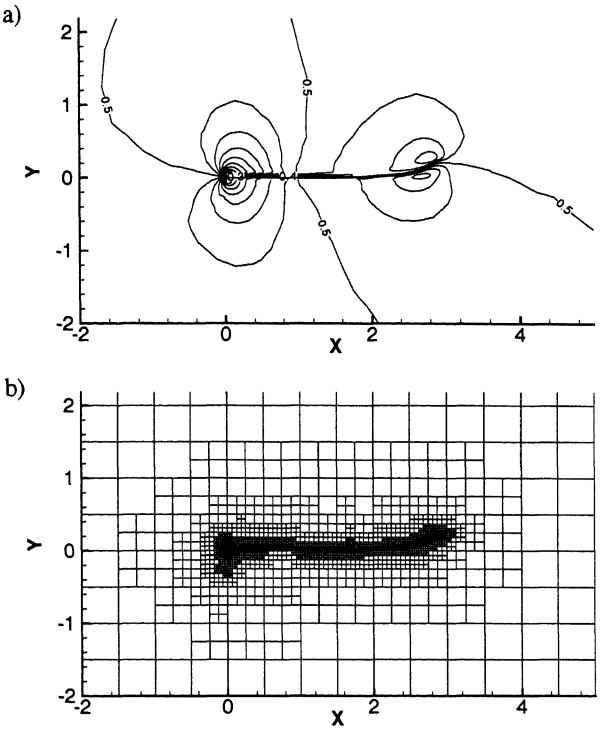

Figure 15: Flat plate, solution at $t = 4$. The flow speed $\sqrt{u^2 + v^2}$ (a) and the adapted grid (b). The plate lies on the x-axis between $x = 0$ and $x = 1$.

RESULTS With the settings found for the refinement criterion, a representative solution is calculated on a large domain (12×12 chord lengths) for $t = 0$ to $t = 9$. As an example, the solution is shown at $t = 4$ in figure 15. At this moment the starting vortex, which was created at the trailing edge from $t = 0$ to $t = 0.75$, has fully developed and moves downstream. The speed distribution in figure 15a shows this vortex and the wake with which it is connected to the plate. Also, the regions of suction and compression, above and below the plate, can be seen. The grid (figure 15b) shows refinements around the plate, the starting vortex and the wake.

A vector plot of the velocity, shown relative to the undisturbed air, clearly shows the starting vortex (figure 16). A region of vorticity is seen, with high speeds around it. Further away from the vortex core, the speed becomes lower. The wake contains vorticity too: the normal components of the velocity vectors do not change across the wake, but the tangential components do. The wake is moved downward by the starting vortex.

From these results, it is concluded that the adaptive gridding method is suitable for the analysis of practical unsteady flow problems. A great reduction of computational costs is achieved: table 2 shows that the computational costs are estimated to be 80 times lower than for a uniform-grid solution.

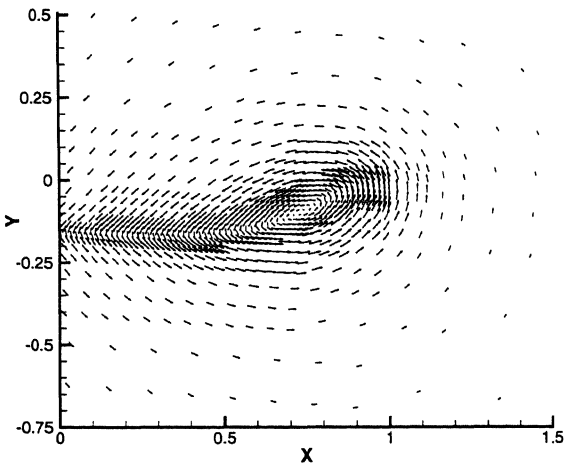

Figure 16: Flat plate, vector plot of the velocity at $t = 4$. In this plot, the undisturbed air is at rest and the flat plate moves (it is now to the left of the y-axis).

	CPU time (s)	% of unif.
uniform grid	121,000	
adapted grid	1448	1.2

Table 2: Flat plate, CPU time for the solution on a refined grid. Large problem, $\alpha = 5^o$. The uniform-grid time is estimated from the CPU times for the smaller convergence problem.

CONCLUSION

Results in the previous section show that the current adaptive gridding method can be used for practical applications and that it is substantially faster than comparable methods without adaptive gridding. The adaptive gridding algorithm gives a very fast and efficient adaptation of the grid to the solution and the entire procedure is easy to implement.

The method is tested here with the 2D Euler equations, but most of the method does not depend on these equations, so the method can be changed easily to solve other hyperbolic conservation laws. Extension to 3D is straightforward too, only the data structure needs some real changes.

Concluding, the current method is a useful technique for a wide range of unsteady problems and an interesting alternative for more complex solution-adaptive methods.

REFERENCES

[1] M.J. Berger and P. Colella. "Local Adaptive Mesh Refinement for Shock Hydrodynamics." J. Comp. Phys. 82 (1989), pp. 64–84.

[2] M.J. Berger and J. Oliger. "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations." J. Comp. Phys. 53 (1984), pp. 484–512.

[3] A.F. Emery. "An Evaluation of Several Differencing Methods for Inviscid Fluid Flow Problems." J. Comp. Phys. 2 (1968), pp. 306–331.

[4] P.W. Hemker and S.P. Spekreijse. "Multiple Grid and Osher's Scheme for the Efficient Solution of the Steady Euler Equations." Applied Numerical Mathematics 2 (1986), pp. 475–493.

[5] W. Hundsdorfer, B. Koren, M. van Loon and J.G. Verwer. "A Positive Finite-Difference Advection Scheme." J. Comp. Phys. 117 (1995), pp. 35–46.

[6] B. Koren. "Multigrid and Defect Correction for the Steady Navier-Stokes Equations, Application to Aerodynamics". CWI Tract 74, Centre for Mathematics and Computer Science, Amsterdam, 1991.

[7] H.T.M. van der Maarel. "A Local Grid Refinement Method for the Euler Equations". PhD Thesis, University of Amsterdam, Amsterdam, 1993.

[8] S. Osher and F. Solomon. "Upwind Difference Schemes for Hyperbolic Conservation laws". Math. Comput. 38 (1982), pp. 339–374.

[9] S.P. Spekreijse. "Multigrid Solution of the Steady Euler Equations". CWI Tract 46, Centre for Mathematics and Computer Science, Amsterdam, 1988.

[10] P.K. Sweby. "High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws." SIAM J. Num. Anal. 21 (1984), pp. 995–1011.

[11] R.A. Trompert. "Local Uniform Grid Refinement for Time-Dependent Partial Differential Equations". PhD Thesis, University of Amsterdam, Amsterdam, 1994.

[12] J. Wackers. "An Adaptive-Gridding Solution Method for the 2D Unsteady Euler Equations". CWI Note MAS-N0301, Centre for Mathematics and Computer Science, Amsterdam, 2003. (Available from www.cwi.nl.)

[13] P.R. Woodward and P. Colella. "The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks." J. Comp. Phys. 54 (1984), pp. 115–173.