Application of a solution-adaptive multigrid method to the Euler equations

H.T.M. van der Maarel, P.W. Hemker, B. Koren[1]
CWI, P.O. Box 4079,
1009 AB Amsterdam, The Netherlands

J.A. Michelsen
Technical University of Denmark
Department of Fluid Mechanics
Bygning 404, 2800 Lyngby, Denmark

A locally refined multigrid method for solving the steady Euler equations of gas dynamics is presented. The method makes use of grids in a locally nested sequence. It is briefly described and next applied to some steady Euler flow problems. The method appears to be more accurate and more efficient than the corresponding multigrid method that applies global refinements only.

1 Introduction

In this paper the adaptive multigrid method as developed at CWI during the first phase of the BRITE/EURAM Area 5 project (contract no. AERO-0003-C), is applied to the Euler equations of gas dynamics. The discretisations on the various levels of refinement introduce a set of nonlinear algebraic equations. The method to solve the set of nonlinear algebraic equations is an application of the nonlinear multigrid scheme, called *full approximation storage* (FAS), possibly embedded in the *full multigrid* (FMG) algorithm or an *iterative defect correction* (IDeC) process. First, these algorithms are described for their use in the present context of locally refined grids. After that, a strategy to introduce local refinements is described. In the experiments to follow, this strategy is used in the grid-refinement cycles. Next, some aspects of the implementation of a local refinement, multigrid method in a computer code are discussed. Finally, a number of examples are presented. The problems chosen for the experiments give an example of the possible range of application, where this method may be used as a tool for analysis of fluid dynamics problems. The problems considered are:

[1] mmmmmmmmnnenrnrnrnrnrnn

- shock reflection on a flat surface;
- transonic flow around an airfoil;
- spurious entropy in the subsonic flow along a compression corner.

First we consider two standard test cases from numerical fluid dynamics to validate the method and to get an idea of possible gain in efficiency of the local grid refinement method with respect to the uniform grid cases. Then, a problem is considered where the method is used to locally introduce a singular grid, in order to approximately solve a problem which has a singular solution, with sufficient accuracy.

In this chapter we also give CPU execution times for the specific implementation of the solution-adaptive, local grid refinement code, run on a typical present-day workstation. We compare these execution times with the execution times for an implementation for uniform grids only, that uses the same multigrid and defect correction algorithms as the adaptive code (cf. Section 5).

Refinement criteria used in all of these experiments are solely based on requirements for the grid in order to provide 'sufficient' resolution for the solution. However, sufficient resolution for the solution does not necessarily imply sufficiently small errors for the discrete approximation of the equations. If one is only interested in the components of the *solution* itself of some problem and not in any of its derivatives, then sufficient resolution depends solely on first-order derivatives of solution components. Apart from first-order derivatives, the local discretisation error usually also depends on higher-order derivatives. Therefore, using only gradients of solution components in the refinement criterion may not be sufficient. The subject of local discretisation errors and their a-posteriori estimation are considered in Chapter 4 of [26].

2 Multigrid and defect correction

2.1 Introductory remarks

The set of algebraic equations obtained by the discretisation introduced in Chapter 2 of [26], is solved by point Gauss-Seidel relaxation, with multigrid convergence acceleration. This particular multigrid procedure is an application of the nonlinear multigrid scheme, called full approximation storage [3]. For the second-order discretisation this process is embedded in an iterative defect correction process [1], [7]. The implementation of the multigrid scheme is directly based on the methods described in [11], [12], [21], [22] and [23], extensively applied in [23], [14] and [16]. Iterative defect correction is described in [1] and [7] and applied in [9], [22], [23], [14] and [16]. The basic method inside the iterative defect correction method, which is used to (approximately) invert the inaccurate discrete operator, is the nonlinear multigrid method.

In this section we give a brief description of the methods, and the slight modifications to our application. The description is a summary of the description presented in [25].

2.2 A locally nested sequence of discretisations

In order to use multigrid we have to specify grid transfer operators. The restriction operators $\overline{R}^l_{l+1}$ and R^l_{l+1} and the prolongation operator P^{l+1}_l are defined such that (i) a sequence of locally nested discretisations on the sequence of locally refined grids is obtained and such that (ii) the coarse-grid equations (see Chapter 2 of [26]) are satisfied implicitly. A locally nested sequence of discretisations $\{N^l\}_{l=0,\ldots,L}$ of the differential operator N is obtained, by definition, if a coarse-grid discrete operator N^l, restricted to the refined cells, is a Galerkin approximation of the fine-grid discretisation. By definition, the restriction $R^{l,l+1}N^l$, as an approximation to N^{l+1}, is called a Galerkin approximation if

$$\left\{N^l(q^l;q^{l-1})\right\}^l_{i,j} = \left\{R^l_{l+1}N^{l+1}(P^{l+1}_l q^l;q^l)\right\}^l_{i,j}, \quad \forall (i,j,l) \in I_f.$$

The restriction operator for the solution, $\overline{R}^l_{l+1} : X^{l+1}(\Omega^{l+1}) \rightarrow X^l(\Omega^l_f)$, is defined by the operator which approximately takes the integral mean value.

$$\left\{\overline{R}^l_{l+1}q^{l+1}\right\}^l_{i,j} = \frac{1}{4}\sum_{m\in K(i,j)} q^{l+1}_m, \quad \forall (i,j,l) \in I_f.$$

This restriction is second-order accurate. For the right-hand side a restriction operator, $R^l_{l+1} : \overline{Y}^{l+1}(\Omega^{l+1}) \rightarrow \overline{Y}^l(\Omega^l_f)$, is defined by

$$\left\{R^l_{l+1}r^{l+1}\right\}^l_{i,j} = \sum_{m\in K(i,j)} r^{l+1}_m, \quad \forall (i,j,l) \in I^l_f. \tag{2.1}$$

The operator for the prolongation of a correction for the solution, $P^{l+1}_l : \overline{X}^l(\Omega^l) \rightarrow \overline{X}^{l+1}(\Omega^{l+1})$, is defined by

$$\left\{P^{l+1}_l q^l\right\}^{l+1}_m = q^l_{i,j}, \quad \forall m \in K(i,j) \text{ and } \forall (i,j) \in I^l_f. \tag{2.2}$$

As shown in [23], [14] and [16], these restrictions and prolongation appear to give very good multigrid performance (together with the point Guass-Seidel relaxation). The prolongation (2.2) and restriction (2.1) satisfy the multigrid rule (cf. [7], [10], [32])

$$m_p + m_r > 2m,$$

where m_p is the order of accuracy of the interpolation used in the prolongation, (for P^{l+1}_l this is $\mathcal{O}(h_l)$), m_r the order of accuracy the restriction (for R^l_{l+1} this is $\mathcal{O}(h^2_l)$, taking into account that R^l_{l+1} is a restriction in $\overline{Y}^l$) and $2m$ the order of the differential equation ($2m = 1$ for the Euler equations). For the given definitions of restrictions and prolongation, the set of first-order accurate discrete equations, exclusive the equations involving a green boundary, form a locally nested sequence of discretisations, (i.e., the coarse-grid discretisation is a Galerkin approximation of the fine-grid discretisation). The first-order accurate reconstruction which uses first-order accurate computation of virtual states (hence first-order weak consistency, cf. Chapter 2 of [26]), yields a locally nested sequence.

2.3 The FAS and FMG scheme

In the nonlinear multigrid algorithm FAS the equations for the first-order accurate discretisation are solved. We identify the discrete operator with first-order accuracy by N_{I}^l, and with second-order accuracy by N_{II}^l. The set of equations to be solved is then given by

$$N_{\mathrm{I}}^l(q^l; q^{l-1}) = s^l, \tag{2.3a}$$

where the right-hand side s^l is given by

$$s_{i,j}^l = \begin{cases} r_{i,j}^l, \quad (i,j,l) \in I_c, \\ \left\{ N_{\mathrm{I}}^l(\overline{R}_{l+1}^l q^l; q^{l-1}) \right\}_{i,j}^l \\ \quad - \left\{ R_{l+1}^l \left(N_{\mathrm{I}}^{l+1}(q^{l+1}; q^l) - s^{l+1} \right) \right\}_{i,j}^l, \quad (i,j,l) \in I_f, \end{cases} \tag{2.3b}$$

and where $r_{i,j}^l$ is defined by

$$r^l = R^l s. \tag{2.4}$$

Upon convergence of the nonlinear multigrid scheme, the solution of (2.3) satisfies

$$\sum_{k \in D} F_{i,j,k}^l(q^l; q^{l-1}, \ldots, q^{l_b}) s_{i,j,k}^l = r_{i,j}^l, \tag{2.5}$$

with appropriate definition of the numerical flux $F_{i,j,l}^l$ and it satisfies

$$q_{i,j}^l = \{ \overline{R}_{l+1}^l q^{l+1} \}_{i,j}^l, \quad \forall (i,j,l) \in I_f. \tag{2.6}$$

The collective, symmetric point Gauss-Seidel relaxation on each level of refinement acts as a smoother in the FAS scheme. For each cell $\Omega_{i,j}^l$ visited, the state $q_{i,j}^l$ is updated, by iterating (through exact Newton iteration) on the local system $\{N_{\mathrm{I}}^l(q^l; q^{l-1})\}_{i,j}^l = r_{i,j}^l$, solving for $q_{i,j}^l$. The residual tolerance for the Newton iteration is taken such that in all but exceptional cases only one or two iterations are performed. The cells on each level are visited in an order which is equivalent to the usual lexicographical order. After a first relaxation sweep has been done, another sweep is done in the reversed direction. This smoother is shown to be very efficient [11], in both subsonic and supersonic Euler flow computations. A FAS cycle, where all q^l, $l = 0, \ldots, L$ are improved, is a recursive algorithm defined by the following steps:

1. improve the solution q^l by applying p pre-relaxations to (2.3a) for level l, resulting in the approximate solution $(q^l)_0$;
2. compute the right-hand side s^{l-1}, determined by (2.3b) for level l;
3. improve the solution q^{l-1} by applying σ FAS cycles to the equations (2.3a) for level $l-1$;

4. compute the correction of the solution, given by the difference of the present coarse-grid solution and the coarse-grid restriction, $d^{l-1} = q^{l-1} - \overline{R}_l^{l-1}(q^l)_0$;

5. improve the solution q^l by adding the prolongation of the coarse grid correction, $q^l = (q^l)_0 + P_{l-1}^l d^{l-1}$;

6. improve the solution q^l by applying q post-relaxation sweeps to the system (2.3a) for level l.

The steps (2)–(5) together are called the coarse grid correction. These steps are skipped for level 0.

The initial solution on the finest level is obtained by application of nested iteration (FMG) [2], [3], [7]. For a level $l \geq 0$, a cycle of the FMG scheme is recursively defined as follows:

1. if $l = 0$ initialise the solution q_0 with some 'arbitrarily' chosen solution; if $l > 0$ initialise the solution on level l with a prolongation $\overline{P}_{l-1}^l q^{l-1}$;

2. improve the solution on level l by application of γ FAS cycles with level l as highest level;

3. if level l is not the highest level, then apply the FMG iteration cycle with a finest level $l + 1$;

Throughout the experiments presented in this chapter we use $\sigma = 1$ (V-cycles), $p = q = 1$ (a single pre-relaxation and a single post-relaxation) and $\gamma = 1$ (a single V-cycle, before starting on a higher level). The prolongation $\overline{P}_l^{l+1}$ used in the FMG algorithm is bilinear interpolation.

2.4 Defect correction

The set of equations for second-order accuracy is solved, using iterative defect correction [7], [1]. The set of higher-order discretised equations on a level l, are given by

$$N_{\mathrm{II}}^l(q^l; q^{l-1}) = r^l. \tag{2.7}$$

The IDeC algorithm solves these equations, by iteratively solving

$$N_{\mathrm{I}}^l(q^l; q^{l-1}) = s^l,$$

applying the FAS scheme, with a modification to the right-hand side s^l for the equations for a cell of the composite grid, i.e. step (2) of the FAS algorithm. An initial solution for the IDeC process is obtained by application of the FAS algorithm to (2.3). In the IDeC iteration the right-hand side s^l depends on the defect of the higher-order accurate equations through

$$s^l_{i,j} = \begin{cases} \left\{ N^l_{\rm I}(q^l; q^{l-1}) \right\}^l_{i,j} - \left\{ N^l_{\rm II}(q^l; q^{l-1}) \right\}^l_{i,j}, & (i,j,l) \in I_c, \\ \left\{ N^l_{\rm I}(\overline{R}^l_{l+1} q^{l+1}; q^{l-1}) \right\}^l_{i,j} & \\ \quad - \left\{ R^l_{l+1} \left(N^{l+1}_{\rm I}(q^l, q^{l+1}) - s^{l+1} \right) \right\}^l_{i,j}, & (i,j,l) \in {\rm I}_f. \end{cases} \tag{2.8}$$

In step (2) of the FAS algorithm the right-hand side is computed by (2.8). Upon convergence of the IDeC scheme (2.7), is satisfied.

In [14] it is shown that one nonlinear multigrid cycle per defect correction cycle is sufficient and most efficient. In all our experiments we do the same and use a single nonlinear multigrid cycle per defect correction cycle.

Before any local grid refinement is introduced, the solution on a basic level l_b is approximately computed. This is done by application of the nested iteration FMG, one or two FAS cycles to approximately solve the first-order accurate discretisation and then a sufficient number of IDeC cycles, for second-order accuracy.

3 Refinement cycles

Solution-adaptive grid refinement involves the grid to be refined at some stage in the solution process. Based on an a-posteriori estimation of relevant quantities appearing in the refinement criterion, the grid is refined where these quantities exceed a pre-set or solution-dependent threshold value, (cf. [4], [20], [19]).

A computation with use of local grid refinement starts with applying the FMG algorithm and possibly subsequent iterative defect correction, so that an approximate solution is obtained for the uniform grid on some basic level l_b. Introduction of local grid refinements is accomplished by the following refinement algorithm, for l the highest level present:

1. determine which cells on level l should be refined, or may be deleted from the system, based on the refinement criterion and an a-posteriori estimation of the relevant quantities used in the refinement criterion and based on the requirement that a virtual state $v^l_{i,j}$ only depends on q^l and q^{l-1};

2. decide whether a grid on level $l+1$ should be created, call the (new) highest level, level L;

3. refine the grid and delete obsolete cells on all levels, from l_b up to and including level $L-1$;

4. initialise the approximate solution of the newly created refinements by application of the prolongation $\widehat{P}^{m+1}_m$, for $m = l_b, \ldots, L-1$ (similar to the FMG algorithm);

5. improve the solution on all levels by application of ρ FAS (first-order discretisation) or IDeC (second-order discretisation) iterations on the composite grid;

6. either apply a refinement cycle on the new system, or solve the present system of equations by a sufficient number of iterations;

The decision in step (2) of the refinement algorithm may be determined by the answer to the question whether the grids on all currently present levels have sufficiently converged, or whether the highest level allowed has already been reached. Notice that for newly created cells, the refinement cycle actually is an application of the nested iteration algorithm FMG, introduced in the previous section. For the prolongation $\widetilde{P}_l^{l+1}$ a bilinear interpolation is used for all newly created cells. In second-order computations, after initialisation of the solution for newly created cells, defect correction is continued, without applying the nonlinear multigrid scheme to the first-order accurate system (2.3) first. The number of iterations ρ before a new refinement cycle is started, step (5) of the refinement algorithm, determines to a large extent the efficiency of the adaptive grid refinement method. However, using an insufficient number of iterations in step (5) may yield a grid too much distorted by the insufficiently converged numerical solution, as compared with the grid that would be obtained with a converged solution. In practice, $\rho = 1$ or $\rho = 2$ for a first-order discretisation appears to yield a grid virtually the same as the grid obtained by using a fully converged solution. For a second-order discretisation $\rho = 4$ or $\rho = 5$ is sufficient.

4 Some aspects of implementation

In order to perform multigrid accelerated Euler flow computations with solution-dependent local grid refinement, a computer code has been developed in portable FORTRAN 77. This code consists of two modules. One module is called BASIS, and is entirely devoted to set up and do maintenance on the data structure. It is described in [13]. The second part, called EULER, consists of all routines related to the adaptive multigrid Euler flow computations. This module is described in [27]. Recently some work has been done on vectorisation of this code for a CRAY Y-MP. This resulted in an additional module, called EUVEL, which is presented in [18].

The data structure reflects the quad-tree relations of the cells in the geometric structure. The grid, made up by the geometric structure, is composed of so-called *patches*. A patch consists of a corner point, and possibly a horizontal wall, a vertical wall and a cell. If a patch contains a cell, sufficient neighbour patches exist to provide the necessary edges and vertices. On the other hand, each corner point, each edge and each cell of the geometric structure belongs to some patch. All data in the structure are stored and referenced through these patches. The patches in the data structure are also related in a quad-tree structure. As a matter of fact, the tree of cells is a subtree of the tree of patches.

In the linked list that implements the quad-tree structure, nine *pointers* are used for each patch. One pointer to the parent of a patch, four pointers to the kid patches and four pointers to the neighbours of a patch. In the FORTRAN implementation the use of pointers in the linked list is emulated by a large, two dimensional array of type *integer*. Each patch has a unique number. For each patch the patch numbers of its parent, its possible kids and its possible neighbours are stored in a row of the integer array. Furthermore, for each patch a set of properties is kept, which identify the *type* of the patch: whether the patch contains a cell, whether the horizontal wall or vertical wall is part of the green boundary or the boundary of the domain, whether the cell contained in the patch may be refined upon the earliest possible occasion, etc.. These properties are stored in a two dimensional array of type *logical*, the column identified by the unique number of a patch and the row identified by a specific property. Finally, all real data for the numerical problem are kept in another two dimensional array of type *real*. These data are also addressed through the unique patch number. For each patch a total of 18 real numbers are stored. The data structure handled by BASIS has a much wider range of applications than Euler flow computations and than cell-centred discretisation schemes.

All actions on the data in the data structure are performed through a depth-first traversal of the tree. The subroutines performing the necessary numerical actions work by application of this tree traversal algorithm.

For each patch visited through this algorithm, a subroutine is called to perform some action on the data in the data structure. The quad-tree structure and the use of such a traversal algorithm to perform any task, is very well suited for the implementation of a multigrid algorithm with adaptive mesh refinement. However, as noted in [18], automatic vectorisation (i.e., vectorisation by the compiler) of a code of this nature does not gain any performance. Therefore, in the vector extension library presented in [18], subroutines are provided that collect pointers to patches containing the geometric structure of a single level of refinement, and place them in an appropriate order in a separate array of pointers. The order of pointers in this array makes the smoothing suited for vectorised processing. Furthermore, in the vectorisable extension the original subroutines that are called for each single patch to perform some numerical action on the data, have been modified so they work on multiple patches. If instead of Osher's numerical flux, Van Leer's numerical flux function is used, (in both the vectorised and non-vectorised code) an overall speed-up of approximately four is obtained.

5 Shock reflection

5.1 The problem

In this section we consider a shock reflection problem. This is a gas dynamics problem of a supersonic flow along a flat solid surface. The domain of definition is $\Omega = \{(x, y) \mid 0 < x < 4, 0 < y < 1\}$, and the flat surface is located at $y = 0$. A shock is impinging from the point $(0, 1)$, at an angle of 29° with the positive x

direction. The boundary conditions for this solution are, at the inflow boundary $x = 0$ given by

$$\begin{aligned} u(0,y) &= 1, \\ v(0,y) &= 0, \\ M(0,y) &= 2.9, \\ \rho(0,y) &= 1. \end{aligned}$$

At the inflow boundary $y = 1$, the flow perpendicular to the horizontal boundary is subsonic, and we impose three conditions. These are approximately given by

$$\begin{aligned} u(x,1) &= 0.90322141, \\ v(x,1) &= -0.17459319, \\ M(x,1) &= 2.37807192. \end{aligned}$$

The boundary $y = 0$ is the solid wall, and we impose impermeability, given by

$$(w \cdot n)|_{y=0} = 0,$$

where n is a normal on the boundary $\partial\Omega$ and $w = (u, v)^T$ the velocity vector. The shock is reflected at the solid wall, at an angle of about $23.279°$. The exact solution is known from simple gas dynamics shock relations. It is a piecewise constant function. The impinging and reflected shocks form the discontinuities of this function.

5.2 Refinement

The domain Ω is rectangular. The coarsest grid used, level zero, is a 6×2 grid. The basic level is $l_b = 1$. Since away from the shock, the exact solution is a constant function, for both, a first-order discretisation and a second-order discretisation, the local discretisation error is zero, away from the shock. For an adaptive computation, it is sufficient *for this problem* to use only the variation of the solution as the refinement criterion. Hence, grids are refined based on the first undivided difference of a component of the solution. According to research on the use of undivided differences as a general refinement criterion, it is found that of any component of the solution, the first undivided difference of density gives best results ([4], [5]).

5.3 Results

For this problem, away from the shock, the discretisation yields equations with local discretisation error equal to zero. The accuracy of the results will be determined to a large extent by the *resolution* provided by the grid used.

5.3.1 First-order discretisation

The equations resulting from the first-order discretisation, are solved on an adaptively refined grid. For the highest level L we take $L = 4$, $L = 5$ and $L = 6$ respectively, to study the convergence behaviour. The number of multigrid FAS iterations (V-cycles) for each refinement cycle is two. A cell is refined if the absolute value of the first undivided difference in either x direction or y direction exceeds 0.05. We consider refinements to have become obsolete if the absolute value of the first undivided difference of density drops under 0.025. In Table. 5.1 the number of cells used are given for both the locally refined and the uniform

TABLE 5.1. Final number of cells used for shock reflection problem; first-order discretisation.

	locally refined		uniform	
L	composite	total	composite	total
4	1533	2040	3072	4092
5	3582	4772	12288	16380
6	7797	10392	49152	65532

grids with a highest level $L = 4, 5, 6$ respectively. Notice that the number of cells approximately doubles, going from finest level $L = L^*$ to finest level $L = L^* + 1$. Figure 5.1 shows for $L = 5$ the grid obtained by local refinement

FIGURE 5.1. Grids and iso-lines of the Mach number for the shock reflection problem on a locally refined and a uniform grid; first-order discretisation; $L = 5$.

and the corresponding uniform grid, with iso-plots of the Mach number on both the adaptive and non-adaptive grid respectively. In Figure 5.2 the convergence histories of both the adaptive method and the method using uniform grids are given. These figures show the logarithm of the mean of the four discrete L_1 norms of the residual of the first-order discretisation, on the components of $\overline{X}_c(\Omega_c)$, for all $(i,j,l) \in I_c$ defined by $(A^l_{i,j})^{-1}\{N^l_{\rm I}(q^l;q^{l-1}) - r^l\}^l_{i,j}$, versus the logarithm of number of Newton iteration steps performed (i.e., the iteration used in the point relaxation to relax the nonlinear system for each cell). Each of the four norms is the discrete L_1 norm of a residual of the discretisation of one of the conservation laws. For $L = 6$ the number of Newton iteration steps to convergence up to machine precision for the adaptive method is about nine times less than the number of iterations needed when a uniform grid is used, while virtually the same solution is obtained (cf. Figure 5.1 and Figure 5.2). For $L = 5$ the number of iterations for the adaptive method is about five times less and for $L = 3$ this is about 2.5 times less.

5.3.2 Second-order discretisation

We use the second-order discretisation $N^l_{\rm II}$, with the Van Albada limiter (cf. [24]), and third-order accurate virtual states as defined in Chapter 2 of [26]. The refinement decision is the same as for the first-order discretisation. The number of defect correction iterations in each refinement cycle is five. It appears that after five defect correction cycles possible wiggles in the 'initial' solution have vanished. The final locally refined grid and iso-lines of Mach number for $L = 5$ are shown in Figure 5.3 The number of cells for local refinement with this second-order discretisation is given in Table 5.2. Notice that the number of cells for

TABLE 5.2. Final number of cells used for shock reflection problem; second-order discretisation.

	locally refined		uniform	
L	composite	total	composite	total
4	924	1228	3072	4092
5	2004	2668	12288	16380
6	4707	6272	49152	65532

all levels $L = 4, 5, 6$ is much smaller for the adaptive computation with the second-order discretisation than for the computation with the first-order discretisation. For the second-order discretisation some extra refinements may be introduced, apart from the refinements introduced by the refinement criterion itself. These are introduced in order to let virtual states for the discretisation on level l, depend only on the solution on levels l and $l-1$.

Convergence histories for locally refined and uniform grids are given in Figure 5.4. This figure shows the logarithm of the mean of the four discrete L_1 norms of the second-order discretisation on $\overline{X}_c(\Omega_c)$, vs. the logarithm of the

FIGURE 5.2. Residual vs. amount of work: convergence histories for adaptively refined and uniform grids; first-order discretisation; $\diamond$: $L = 4$; $+$: $L = 5$; $\square$: $L = 6$; ———: locally refined; — —: uniform.

FIGURE 5.3. Grid and iso-lines of the Mach number for the shock reflection problem on a locally refined grid; second-order discretisation; $L = 5$.

FIGURE 5.4. Residual vs. amount of work: convergence histories for defect correction and second-order discretisation on uniform and locally refined grids; $\diamond$: $L = 4$; $+$: $L = 5$; $\square$: $L = 6$; ———: locally refined; — —: uniform.

TABLE 5.3. CPU-time required per Newton iteration step for the shock reflection problem.

	adaptive-grid code		uniform-grid code
	locally refined	uniform	
FAS	0.96 ms/iter.	0.93 ms/iter.	0.84 ms/iter.
IDeC/FAS	1.14 ms/iter.	1.03 ms/iter.	0.84 ms/iter.

number of Newton iteration steps. We did not consider $L = 6$ and a uniform grid. This problem is so large that it causes the computer to start swapping pieces of memory to disk, resulting in a very large processing time. Apparently, the defect correction process does not converge for uniform grids. The reason for this is possibly the following. On a uniform grid, with finest level L, many more Fourier modes can be represented than on the refined grid with finest level L. Especially low-frequency modes can be represented very well on the uniform grid, better than on the locally refined grid. In [6] an amplification factor $g \approx 1$ for low-frequency Fourier modes is found, in case of the linear convection problem in two space dimensions. However, it should be stressed that for this linear convection problem this high amplification factor corresponds to functions that are constant in the characteristic direction of the problem.

The defect correction algorithm for the locally refined grids does converge. For the second-order discretisation and defect correction, the discretisation on a locally refined grid yields a more robust algorithm for this problem.

5.4 Execution time

In order to get some idea of execution time, for this problem we give CPU-times of our FORTRAN research code on an SGI IRIS INDIGO XS workstation. All optimisation was done by the compiler. We give the average CPU-time it takes for all operations of the local refinement computation, per Newton iteration step. Note that the Newton iterations referenced here are *local* Newton iterations, used in the point relaxations. The results are given in Table 5.3. This table also shows the average CPU-time for another multigrid code, developed to work with uniform grids only. This particular code, called EULER7, implements the same multigrid and defect correction algorithms as used in the code for adaptive computations (cf. [14]). The FAS algorithm on a locally refined grid appears to be only three percent more expensive per Newton iteration step than on a uniform grid with the adaptive code. The iterative defect correction for a locally refined grid appears to be about 18% more expensive per Newton iteration step than iterative defect correction on a uniform grid. For the FAS algorithm, the adaptive code *with* local grid refinement, appears to be about 14% more expensive per Newton iteration step, than the non-adaptive code EULER7. For iterative defect correction, the adaptive-grid code is about 34% more expensive per Newton iteration step than EULER7.

6 Airfoil flow

In this section we consider the transonic flow around the NACA0012-airfoil. The flow conditions at the far-field boundary are: $M_\infty = 0.8$, angle of attack $\alpha = 1.25°$, $\rho_\infty = 1$ and $(u^2 + v^2)_\infty = 1$. The computational domain extends to about 100 chords to all sides.

As the second-order operator N^l_{II}, we use the Van Albada limiter scheme [24], [21] and, again, third-order accurate computation of virtual states (cf. Chapter 2 of [26] for details). The limiter scheme is used, since spurious wiggles in the solution are expected if a second-order non-limiter scheme is used.

In the refinement criterion we use first undivided differences of the density, in both streamwise direction and perpendicular to the stream line direction. Two thresholds are used, one for each direction. This prevents the algorithm from refining in the neighbourhood of a shock only. It allows the algorithm also to find the contact discontinuity, and to resolve the expansion region. Then, we not only get a good resolution of the shock, but also a good resolution of the expansion, and this in turn is important for the accurate computation of the lift and drag coefficients. The use of only one threshold value (i.e., the same threshold for both criteria) would be inefficient for a small threshold value (too many refinements) On the other hand, a larger threshold value only refines at strong discontinuities.

The grid used is an O-type grid. The coarsest grid is a 5×8 grid. The highest level is $L = 5$. The uniform grid for level $l = 1$ is shown in full and in detail in Figure 6.1. A cell is refined if the first undivided difference of density

Figure 6.1. Uniform grid of level $l = 1$, around NACA0012 airfoil.

in flow direction is larger than 0.02, or when this difference in the direction

perpendicular to the flow is larger than 0.004. The final adaptively refined grid, with $L = 5$, is shown in Figure 6.2. In Figure 6.3 the Mach number

FIGURE 6.2. Locally refined grid with $L = 5$, around NACA0012 airfoil.

FIGURE 6.3. Iso-line plots of the Mach number for the transonic flow around a NACA0012 airfoil; $\alpha = 1.25°$; $M_\infty = 0.8$; locally refined grid: $L = 5$; uniform grid: $L = 4$.

FIGURE 6.4. Pressure distribution for adaptively refined and uniform grids for transonic flow around a NACA0012 airfoil; $\alpha = 1.25°$; $M_\infty = 0.8$; ———: locally refined grid, $L = 5$; — —: uniform grid, $L = 4$.

distributions are shown both for an adaptively refined and for a uniform grid. The pressure distribution for the uniform grid and for the locally refined grid are shown in Figure 6.4. For the lift and drag coefficient on the adaptively generated composite grid we find $c_l = 0.3480$, $c_d = 0.0235$. On the non-adaptive grid we find $c_l = 0.3512$ and $c_d = 0.0235$. The difference between these values is less than 10% of the scatter found between different reference results listed in [31]. This reference gives $c_l = 0.3632$ and $c_d = 0.0230$ obtained on a grid of 20480 cells, by SCHMIDT and JAMESON [31]. The number of cells on the adaptively generated composite grid is 7876 and a total number of 10488 cells was used in the computation. The non-adaptive grid uses 10240 cells on the finest grid and a total number of cells of 13640. The convergence histories of both the adaptive and non-adaptive case are shown in Figure 6.5. The adaptive computation takes about three times less work than the computation on the non-adaptive grid.

7 SPURIOUS ENTROPY GENERATION FOR SUBSONIC FLOW PAST A COMPRESSION CORNER

7.1 Introduction

In this section we study the numerical entropy generation for the steady, two dimensional Euler equations and a perfect gas. Numerical approximations of the subsonic Euler flow along a compression corner show spurious entropy generation, which is virtually independent of the mesh size of the computational grid. Sometimes, simple incompressible flow models are used to describe the flow in

FIGURE 6.5. Residual vs. amount of work: convergence histories of defect correction and second-order discretisation for NACA0012 airfoil flow; $\alpha = 1.25°$; $M_\infty = 0.8$; locally refined grid: $L = 5$; uniform grid: $L = 4$.

the vicinity of geometric singularities, such as the compression corner shown in Figure 7.1. The presumption that the velocity near the corner is small, is then used to justify incompressible wedge flow as a model. The compressibility effect is often accounted for by a correction, such as the Prandtl-Glauert rule. However, in [30] it is found analytically, using the hodograph transformation, that even in a subsonic case the flow does not have to stagnate in the corner. Hence, the use of an incompressible model to approximate the subsonic compressible flow along a compression corner may be unrealistic, since for incompressible models the corner is a stagnation point. Furthermore, the singularity in the solution at the corner for compressible flow appears to be considerably more complex than that for incompressible flow. Trying to remove the singularity by postulating the same singularity as for incompressible potential flow (i.e., an inverse power of the distance to the corner) in the Euler model, has proved to be unsuccessful (cf. [29] and [28]).

In a numerical method, discrete convergence of the solution may be obtained by considering the compression corner as the limit of a smooth surface, which is parametrised by the mesh width of the grid. However, convergence is slow. This problem shows a way to use a local grid refinement technique as a tool to approximate a singular solution. For the approximation of a singular solution, the grid becomes singular too, for mesh width $h_l \to 0$. We use the solution-adaptive grid refinement method to obtain reasonable discrete convergence, without excessive computational cost.

FIGURE 7.1. Subsonic flow along a kinked wall.

7.2 The problem

A typical layout of the geometry of the problem at hand is shown in Figure 7.1, and the corner with angle δ. We consider a flow problem which is known to be *exactly* homentropic. As is well-known, in steady subsonic flow, the entropy along a stream line is constant. Hence, a first requirement for obtaining a homentropic flow is to impose a constant entropy at inflow in the domain. Furthermore, at the inflow boundary the velocity vector is imposed. This velocity corresponds with that of an incompressible potential flow (i.e. irrotational) along the surface, which is analytically known by conformal mapping. At outflow the corresponding pressure is imposed.

Boundary conditions are incorporated into the discretisation scheme in a way which is consistent with the discretisation in the interior of the computational domain (cf. [12]). In subsonic flows, this requires *three* boundary conditions at inflow and *one* boundary condition at outflow. Notice that by just obeying these numbers in subsonic flow computations, mathematical well-posedness is not yet guaranteed. For a study of the mathematical well-posedness of some typical subsonic outflow boundary conditions, we refer to [15].

The domain of definition is the area covered by the grid in Figure 7.2. With the grid shown in Figure 7.2 and similar 16×16 and 8×8 grids, a straightforward application of the discretisation gives for the entropy a result as shown in Figure 7.3. This clearly shows that the entropy error is virtually independent of the refinement of the grid.

7.3 Nature of the error

In [28] a number of possible causes for this behaviour are considered, in order to make the nature of the zeroth-order error plausible.

First the local discretisation error is studied. In the discretisation an inconsistency is encountered, which is due to the kink in a grid as in Figure 7.2. This inconsistency in the discretisation appears in equations derived for the cells directly bordering the grid line emerging from the corner. It appears from numerical experiments with the kinked grid and a smooth flow, that this inconsistency has no adverse effect on the entropy error.

FIGURE 7.2. The uniform grid along a compression corner; 32×32.

FIGURE 7.3. The entropy error for the computation of the flow past a compression corner; $\delta = 10°$; $\diamond$: 8×8; $\square$: 16×16; $+$: 32×32.

Secondly, the discretisation of the solid-wall boundary conditions are studied, again by numerical experiment. For this purpose, the wedge-shaped wall is replaced by a continuously curved wall. The boundary conditions are discretised identically as in the case of the wedge shaped wall. It appears that the discretisation of the solid wall boundary conditions also has no adverse effect on the entropy error.

From the experiments it becomes plausible that the singularity in the exact solution itself causes the bad convergence behaviour.

7.4 Parametrised smooth wall

In this section we study the entropy error in the flow along a continuously curved wall. The shape of the wall that we use, is given by

$$y_w(x) = \begin{cases} 0, & x \le -\frac{1}{2}l, \\ \left(\frac{(x+\frac{1}{2}l)^3}{l^2} - \frac{1}{2}\frac{(x+\frac{1}{2}l)^4}{l^3} \right) \tan\delta, & -\frac{1}{2}l < x \le \frac{1}{2}l, \\ x \tan\delta, & \frac{1}{2}l < x. \end{cases} \tag{7.1}$$

Here, l is the length of the curved part of the wall and δ the angle between the positive x direction and the uncurved part of the wall at $x > \frac{1}{2}l$. The geometry is shown in Figure 7.4. The wall is defined in such a way that $y_w \in C^2[-1, \cos\delta]$.

FIGURE 7.4. The flow along a continuously curved wall.

The grid used is shown in Fig 7.5. In Figure 7.6, for $\delta = 10^\circ$ and $l = 1$ the entropy error along the wall is shown for the 32×32, 16×16 and 8×8 grid. We find that the entropy error is first-order in mesh size as already mentioned in Section 7.3.

In these results the length l of the continuously curved wall segment is the same for all grids considered. Now we re-compute the flow along a continuously curved wall (7.1), but we let l depend on the mesh size. We use $l = \mathcal{O}(h^p)$, $p > 0$, where h is the mesh width in the direction of the x axis. If $p < \infty$, it is clear that for $h \to 0$ the curved wall degenerates into the wedge-shaped wall. The results in Figure 7.3 and in Figure 7.6 can be considered as those for the

FIGURE 7.5. A grid along a continuously curved wall; 32×32

FIGURE 7.6. The entropy error for the computation of the flow past a continuously curved wall; $\delta = 10°$; $\diamond$: 8×8; $\square$: 16×16; $+$: 32×32.

limit $p = \infty$ and $p = 0$, respectively. The number of cells N along the curved part of the wall is

$$N = \frac{l}{h}.$$

Hence, with $l = \mathcal{O}(h^p)$, we have

$$N = \mathcal{O}(h^{p-1}).$$

For $p \geq 1$ and in the limit $h = 0$, there would be only one grid line emerging from the corner, and we arrive at a similar situation as for $p = \infty$ (i.e., zeroth-order entropy error). Thus, looking for decreasing entropy errors for decreasing mesh width, solving the problem of subsonic flow past a compression corner, we must take $0 < p < 1$. In Figure 7.7 the behaviour of the entropy error when

FIGURE 7.7. The order of discrete convergence; $l = h^p$; $\delta = 10°$; $\diamond$: computed from 16×16 and 32×32 grids; $\square$: computed from 32×32 and 64×64 grids.

l decreases as a function of h. Here, for l we take $l = c_1 h^p$, with c_1 constant, and for the entropy error we assume the form

$$\left\| \frac{s}{s_{\text{ref}}} - 1 \right\|_\infty = c_2 h^q,$$

c_2 constant, and $\|.\|_\infty$ a discrete supremum norm. From numerical experiments with $c_1 = 1$, on a 16×16, a 32×32 and a 64×64 grid, q has been determined as a function of p. As already shown, for $p = 0$ we find q approaches 1, as h approaches 0.

We find that for a subsonic flow, the flow along a curved wall can be used to successfully compute the flow along a compression corner. For $p \in (0, 1)$, and

$h \to 0$, the curved wall becomes kinked and the entropy error vanishes, because for any $p \in (0,1)$ it appears that $\|s/s_{\text{ref}} - 1\|_\infty = \mathcal{O}(h^q)$, and $q > 0$. If we want to have the entropy error disappear at the same rate as l, then according to Figure 7.7, we should take $p \approx 0.4$. In Figure 7.8 we give the entropy error

FIGURE 7.8. The entropy error for the computatrion of the flow past a continuously curved wall; $l = h^{0.4}$; $\delta = 10°$; $\diamond$: 16×16; $\square$: 32×32; $+$: 64×64.

distributions along the wall, as obtained for $p = 0.4$ on a 16×16, a 32×32 and a 64×64 grid. Given the rather low order of accuracy, $q(p = 0.4) \approx 0.4$, reduction of the entropy error below some required tolerance level may become expensive when applying *uniform* grid refinement. The remedy to this lies in the application of *local* grid refinement. As an example, in Figure 7.9 we give results obtained on a 32×32 grid and local refinements, with $p = 0.4$. The entropy error is used in the refinement criterion. The criterion is based on the discrete gradient of the entropy error in stream line direction $(v \cdot \nabla s)/|v|$, multiplied by the square root of the area of a cell, and with v the velocity vector, s the entropy and ∇ a discrete gradient operator. Figure 7.10 gives the locally adapted grid for the result of Figure 7.9: the 32×32-grid with four additional levels of local refinement is shown. Notice that for decreasing mesh width, the refined regions get smaller and closer to the corner. Finally, in Table 7.1, for the three grids considered in Figure 7.9 we give an impression how the rounded corner converges to the kink for decreasing mesh width.

7.5 Concluding remarks

It is possible to remove the zeroth-order global discretisation error from the numerical approximation of the subsonic Euler flow past a compression corner.

FIGURE 7.9. The entropy error for computation of flow past a continuously curved wall with adaptively refined grids; $l = h_l^{0.4}$; $\diamond$: 32×32; $\square$: 32×32 with two levels of refinement; $+$: 32×32 with four levels of refinement.

FIGURE 7.10. Adaptively refined grid; $l = h_l^{0.4}$; $\delta = 10°$; 32×32 grid, with four levels of local refinement.

TABLE 7.1. Geometrical data for the adaptively refined grids, with $l = h_l^{0.4}$, $\delta = 10°$.

grid	l	N
32×32 without local refinement	0.3299	5
32×32 with two levels of local refinement	0.1895	12
32×32 with four levels of local refinement	0.1088	28

Poor computational efficiency due to the rather low order of accuracy, may be effectively circumvented by application of local, solution-adaptive grid refinement. Numerical results indicate that local refinement (combined with the smooth discretisations of the kinked wall) *is* an alternative to reduce the error, without increasing the computational effort excessively .

We found the paradoxical result that by making a sequence of geometrically less accurate discretisations of the compression ramp, a numerical solution of the flow with better error behaviour can be obtained. The less accurate discretisations of the kinked wall employ discrete smooth versions of the exact kinked wall. By making the discretisation of the geometry dependent on the mesh size h, an $\mathcal{O}(h^q)$, $0 < q < 1$, entropy error can be obtained. For this result, the grid has a singularity at the compression corner.

The solution-adaptive multigrid method appears to be a suitable tool for detailed studies of other singular flow phenomena as well (cf. [17]).

REFERENCES

1. K. BÖHMER, P.W. HEMKER, H.J. STETTER (1984). The defect correction approach, *Defect Correction Methods, Computing, Suppl. 5* (K. BÖHMER and H.J. STETTER, eds.), Springer Verlag, Wien, pp. 1–32.
2. A. BRANDT (1980). Multilevel adaptive computations in fluid dynamics, *AIAA J.* **18** 1165–1172.
3. _____(1982). Guide to multigrid development, *Multigrid Methods, Lecture Notes in Mathematics* (W. HACKBUSCH and U. TROTTENBERG, eds.), Springer-Verlag, pp. 220–312.
4. J.F. DANNENHOFFER (1987). III, *Grid adaptation for complex two-dimensional transonic flows*, Ph.D. thesis, Massachusetts Institute of Technology.
5. _____(1989). Adaptive grid embedding for complex two-dimensional flows, *Adaptive Methods for Partial Differential Equations* (Troy, 1988) (J.E. FLAHERTY, P.J. PASLOW, M.S. SHEPHARD, J.D. VASILAKIS, eds.), Rensselaer Polytechnique Institute, Society for Industrial and Applied Mathematics, Philadelphia, pp. 68–82.
6. J.-A. DESIDERI (1990). Analysis of the convergence of iterative implicit and defect-correction algorithms for hyperbolic problems, *Report NM-R9004*, CWI.

7. W. HACKBUSCH, (1985). *Multi-Grid Methods and Applications*, Springer Series in Computational Mathematics, vol. 4, Springer-Verlag, Berlin.
8. W. HACKBUSCH, U. TROTTENBERG (eds.) (1986). Multigrid Methods II, *Proc. of the 2nd European Conference on Multigrid Methods, held in Cologne*, 1985, Lecture Notes in Mathematics, vol. 1228, Cologne, 1985, Springer-Verlag.
9. P.W. HEMKER (1986). Defect correction and higher order schemes for the multi grid solution of the steady Euler equations, In HACKBUSCH and TROTTENBERG [8], pp. 149–165.
10. ——— (1990). On the order of prolongations and restrictions in multigrid procedures, *J. Comput. Appl. Math.* **32**, 423–429.
11. P.W. HEMKER, S.P. SPEKREIJSE (1985). Multigrid solutions of the steady Euler equations, *Advances in Multi-Grid Methods, Proc. conference held in Oberwolfach, Notes on Numerical Fluid Mechanics, vol. 11* (Oberwolfach, 1984) (D. Braess, W. Hackbusch, and U. Trottenberg, eds.), Vieweg Braunschweig, pp. 33–44.
12. ——— (1986) Multiple grid and Osher's scheme for the efficient solution of the steady Euler equations, *Appl. Num. Math.* **2**, 475–493.
13. P.W. HEMKER, H.T.M. VAN DER MAAREL, C.T.H. EVERAARS (1990). BASIS: A data structure for adaptive multigrid computations, *Report NM-R9014*, CWI, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands.
14. B. KOREN (1988). Defect correction and multigrid for the efficient and accurate computation of airfoil flows, *J. Comput. Phys.* **77**, 183–206.
15. ——— (1989). Euler flow solutions for transonic shock wave – boundary layer interaction, *Internat. J. Numer. Methods Fluids* **9**, 59–73.
16. ——— (1990). *Multigrid and Defect Correction for the Steady Navier-Stokes Equations, Application to Aerodynamics*, CWI, Amsterdam, CWI-tract 74.
17. B. KOREN H.T.M. VAN DER MAAREL (1992). On steady, inviscid shock waves at continuously curved, convex surfaces, *Report NM-R9202*, CWI.
18. W.M. LIOEN M. LOUTER-NOOL (1993). *EUVEL: An EULER vector extension library*, In preparation.
19. J.A. MICHELSEN (1991). Investigation of solution-derivative based adaption criteria for inviscid supersonic flow over bump, *Report Technical University of Denmark AFM Aero 0003-01-TUD.*
20. K.G. POWELL, M.A. BEER, G.W. LAW (1989). An adaptive embedded mesh procedure for leading-edge vortex flows, *AIAA Paper 89–0080.*
21. S.P. SPEKREIJSE (1986). Multigrid solution of monotone second-order discretizations of hyperbolic conservation laws, *Math. Comp.* **49**, 135–155.
22. ——— (1986). Second order accurate upwind solutions of the 2D steady Euler equations by the use of a defect correction method, In HACKBUSCH and TROTTENBERG [8], pp. 285–300.
23. ——— (1988). Multigrid Solution of the Steady Euler Equations, CWI, Amsterdam, CWI-tract 46.
24. G.D. VAN ALBADA, B. VAN LEER, W.W. ROBERTS (1982). A comparative study of computational methods in cosmic gas dynamics, *Astron. Astrophys.*

108, 76–84.
25. H.T.M. VAN DER MAAREL (1992). Adaptive multigrid for the steady Euler equations, *Comm. Appl. Numer. Methods* **8**, 749–760.
26. _____(1993). A local grid refinement method for the Euler equations, Ph.D. thesis, University of Amsterdam.
27. H.T.M. VAN DER MAAREL, P.W. HEMKER C.T.H. EVERAARS (1990). EULER: An adaptive Euler code, *CWI report NM-R9015.*
28. H.T.M. VAN DER MAAREL, B. KOREN (1991). Spurious, zeroth-order entropy generation along a kinked wall, *Internat. J. Numer. Methods Fluids* **13**, 1113–1129.
29. A. VERHOFF (1991). *Private communication.*
30. A. VERHOFF, D. STOOKESBERRY, T. MICHAL (1991). Hodograph solution for compressible flow past a corner and comparison with Euler numerical predictions, *Tech. Rep. MCAIR91-005, McDonnell Aircraft Company*, Saint Louis, MO.
31. H. VIVIAND (1985). Numerical solutions of two-dimensional reference test cases, *AGARD-AR-211*, AGARD.
32. P. WESSELING (1991). *An Introduction to Multigrid Methods*, John Wiley and Sons, Ltd., Chichester.