Chapter 5

Multi-Grid on Unstructured Grids

One of the desired objectives of the *MOUSE* library is to be a flexible and open development platform for numerical simulations. From the implementation point of view, modern programming languages offer powerful tools for flexibility, as for example the inheritance of object oriented programming or *C++* 's template mechanism (see chapter 6). The numerical approach, however, should be flexible as well. This chapter describes the numerical aspects of an FAS multi-grid algorithm build in such a context to improve the numerical efficiency. A full approximation scheme (FAS) has been the choice, because it provides the highest flexibility in terms of the actual equation to be solved (described in [18]).
To keep such a platform open for new developments, it is a major design goal to decouple the multi-grid implementation as much as possible from specialized implementations related to the problems to be solved. Ideally it should be possible to run a multi-grid application with just the same source code that has once been written to run a simple explicit iteration. To achieve this a simple explicit scheme shall be employed for the smoothing iterations on each grid level.
A more classical approach tries to formulate implicit instead of explicit schemes. Implicit schemes linearize the non-linear equations to be solved, around a certain point in time. This linearized equation can then be discretized on the new time level, and leads to a coupled system of linear equations. Schemes of this type can use large time steps, since an implicit scheme is unconditionally stable. Multi-grid will then be used to accelerate the linear solver, which has to be used to advance to a new time level. Due to the linearity of the equations which will be solved by the multi-grid scheme, the behavior of the scheme becomes more predictable, which is a big advantage. These methods are known to be numerically efficient and robust. Unfortunately, new physical problems or new discretizations require a lot more implementation work, which is dependent on the equation to be solved. Furthermore they might lead to high memory requirements. Consequently, explicit methods have been chosen for this open development platform.
Another important point is the generation of different grid levels for the multi-grid process. The volume agglomeration technique has been chosen to construct a sequence of meshes. This technique as presented in [19] is considered as a black box method for generating finite volume coarse levels. Some more recent works have proven to be efficient in

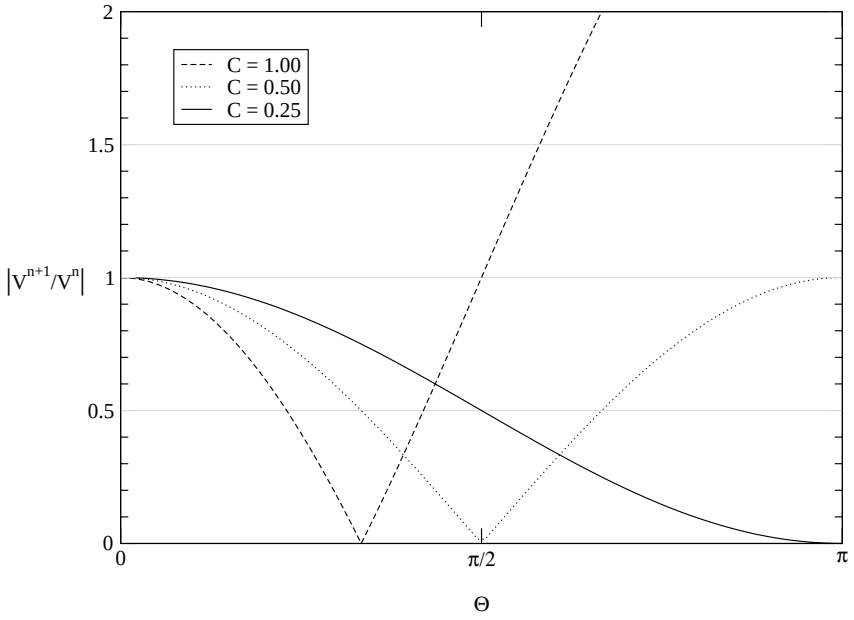

Figure 5.1: Amplification for a simple iteration-scheme.

the field of computational fluid dynamics [20].
If used together, the FAS based on explicit discretizations and the volume agglomeration technique, are close to be a black box multi-grid algorithm. Hence it could be used to accelerate the convergence of various problems. The main problems as well as the approaches to solve them overcome those will be presented in this section. For all parts of the algorithm, flexibility in terms of the equations to be solved, as well as the discretization used, was a major guideline, although not fully achieved by now.
This chapter will briefly explain the basic idea behind the multi-grid method. A quick explanation of a linear multi-grid will then be given, before the FAS method and its implementation will be described.

5.1 Basic Idea of Multi-Grid Methods

Consider the following linear equation:

$$\frac{\partial^2 \phi}{\partial x^2} = 0\,. \tag{5.1}$$

To solve this one dimensional elliptic model-equation, a simple finite difference method can be used. In order to demonstrate the behavior of different iteration methods, (5.1) shall be written as a time dependent problem:

$$\frac{\partial \phi}{\partial t} - \frac{\partial^2 \phi}{\partial x^2} = 0\,. \tag{5.2}$$

Heat conduction, for example, can be described by an equation of this type. A simple finite difference approximation of (5.2), using a simple forward difference for the time

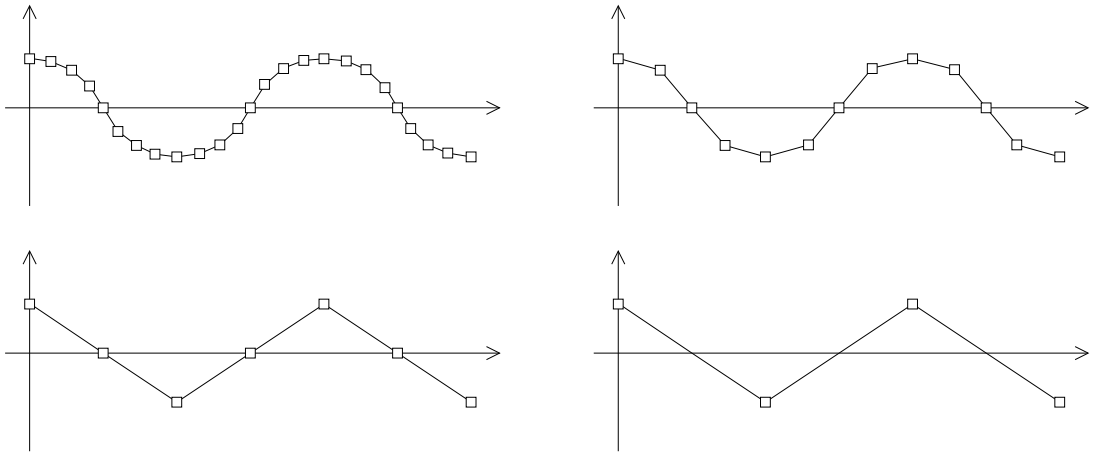

Figure 5.2: A discrete wave on different grid levels.

derivative, reads:

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} - \frac{\phi_{i-1}^n - 2\phi_i^n + \phi_{i+1}^n}{h^2} = 0\,, \tag{5.3}$$

where h is the spatial step-size, which is constant for the whole domain. A value for ϕ on a new time level can be computed, using the following rule:

$$\phi_i^{n+1} = \phi_i^n + C(\phi_{i-1}^n - 2\phi_i^n + \phi_{i+1}^n) \quad , \quad C = \frac{\Delta t}{h^2}\,. \tag{5.4}$$

For $C = \frac{1}{2}$ the standard Jacobi iteration will be obtained:

$$\phi_i^{n+1} = \frac{\phi_{i-1}^n + \phi_{i+1}^n}{2}\,. \tag{5.5}$$

The Jacobi iteration is known to have a quadratic complexity. This means to reduce the norm of the error by a constant factor, four times more iterations are needed, if the step-size h will be reduced by a factor of two. Multi grid methods try to reduce this complexity, since other point-wise iteration do not have better complexities as well. To illustrate the problems of a single-grid iteration, the exact solution $\boldsymbol{\phi}$ shall now be decomposed into an approximation $\boldsymbol{\phi}^n$ and a corresponding error $\boldsymbol{\phi}_{\mathrm{err}}^n$:

$$\boldsymbol{\phi}_{\mathrm{err}}^n = \boldsymbol{\phi} - \boldsymbol{\phi}^n \quad , \quad \boldsymbol{\phi} = \lim_{n\to\infty}(\boldsymbol{\phi}^n)\,. \tag{5.6}$$

The linearity of (5.2) allows the update rule (5.4) to be written for the error

$$\boldsymbol{\phi}_{\mathrm{err,i}}^{n+1} = \boldsymbol{\phi}_{\mathrm{err,i}}^n + C(\boldsymbol{\phi}_{\mathrm{err,i-1}}^n - 2\boldsymbol{\phi}_{\mathrm{err,i}}^n + \boldsymbol{\phi}_{\mathrm{err,i+1}}^n) \quad , \quad C = \frac{\Delta t}{h^2}\,. \tag{5.7}$$

This error can be written as a Fourier series, containing all wave-lengths that can be reproduced on a given set of discrete nodes:

$$\boldsymbol{\phi}^n_{\mathrm{err,i}} = \sum_{k=k_{\min}}^{k_{\max}} V^n(k)\, e^{Iki\Delta x} \quad , \quad k = \frac{2\pi}{\lambda}\,,\; I = \sqrt{-1}\,, \tag{5.8}$$

with k is the wave-number and $V^n(k)$ as the amplitude belonging to this wave-number. The minimally resolvable wave-length is $\lambda_{\min} = 2\Delta x$ and the maximum is $\lambda_{\max} = 2L$, with L being the length of the integration interval. By introducing the angle $\Theta = k\Delta x$, the periodic error can be written as:

$$\boldsymbol{\phi}^n_{\mathrm{err,i}} = \sum_{k=k_{\min}}^{k_{\max}} V^n(\Theta)\, e^{I\Theta}\,. \tag{5.9}$$

The ratio between the amplitude of a new time level ($V^{n+1}(\Theta)$) and the amplitude of the current time level ($V^n(\Theta)$) describes the amplification of the wave-length corresponding to the angle Θ. For (5.4) the amplification factor can be easily computed:

$$\frac{V^{n+1}(\Theta)}{V^n(\Theta)} \;=\; 2\,C\,(\cos\Theta - 1) + 1\,. \tag{5.10}$$

Figure 5.1 shows the amplification factor versus the angle Θ for different values of C. Stable schemes require a value of $C < \frac{1}{2}$. As it can be seen in this figure, schemes do not provide a high damping of disturbances for all wave-lengths. For long waves (5.4) is not efficient. The idea of the multi-grid method is to have an efficient iteration scheme for all possible wave-lengths. This can be done by switching between different grids. If a long wave disturbance is transferred from one grid to a coarser one, the discrete wave-lengths become shorter. Figure 5.2 illustrates how a long wave becomes shorter, while sequentially transferred to coarser grids.

5.2 Linear Multi-Grid

This paragraph explains how a multi-grid scheme can be derived to solve linear equations. Any set of linear equations can be written as follows:

$$\mathrm{L}(\mathbf{q}) \;=\; \mathbf{b}\,. \tag{5.11}$$

L denotes the linear operator, and (5.11) can be discretely represented on a grid with the constant step-size h:

$$\mathrm{L_h}(\mathbf{Q_h}) \;=\; \mathbf{B_h}\,. \tag{5.12}$$

The index h denotes a discretization on a grid with h as step-size. Given any approximation $\mathbf{Q}^k_{\mathrm{h}}$ during the iteration process, the following can be used to define the residual (remainder) of (5.12):

$$\mathrm{L_h}(\mathbf{Q}^k_{\mathrm{h}}) \;+\; \mathbf{R}^k_{\mathrm{h}} \;=\; \mathbf{B_h} \tag{5.13}$$

Please note that this residual is not the spatial operator Res that has been previously defined. An error, representing the difference between the exact solution of (5.12) and

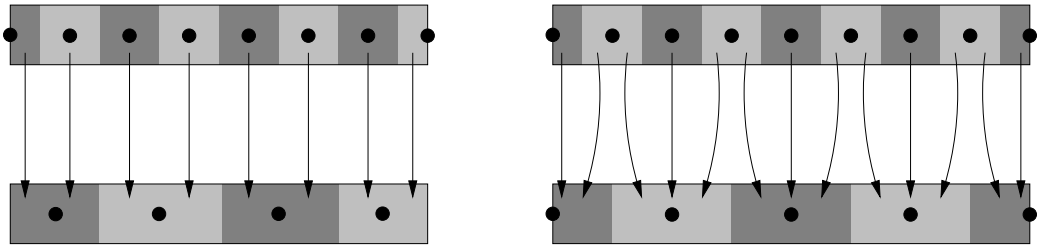

Figure 5.3: Construction of nested(left) and non-nested(right) coarse levels.

any approximation during the iteration process, can be defined. With the exact solution of the discretized equation (5.12):

$$\mathbf{Q}_{\mathrm{h}}^{\infty} = \lim_{k\to\infty}(\mathbf{Q}_{\mathrm{h}}^{k}), \tag{5.14}$$

the error of an iteration level k can be defined as:

$$\mathbf{Q}_{\mathrm{h,err}}^{k} = \mathbf{Q}_{\mathrm{h}}^{\infty} - \mathbf{Q}_{\mathrm{h}}^{k} \quad \Leftrightarrow \quad \mathbf{Q}_{\mathrm{h}}^{k} = \mathbf{Q}_{\mathrm{h}}^{\infty} - \mathbf{Q}_{\mathrm{h,err}}^{k} \,. \tag{5.15}$$

Using the linearity of L_h the following superposition is possible:

$$\mathrm{L_h}(\mathbf{Q}_{\mathrm{h}}^{\infty} - \mathbf{Q}_{\mathrm{h,err}}^{k}) = \mathrm{L_h}(\mathbf{Q}_{\mathrm{h}}^{\infty}) - \mathrm{L_h}(\mathbf{Q}_{h,err}^{k}), \tag{5.16}$$

and an equation for the error $\mathbf{Q}_{\mathrm{err,h}}$ can be derived by combining (5.13) and (5.15):

$$\begin{aligned} & \mathrm{L_h}(\mathbf{Q}_{\mathrm{h}}^{\infty} - \mathbf{Q}_{\mathrm{err,h}}^{k}) + \mathbf{R}_{\mathrm{h}}^{k} = \mathbf{B}_{\mathrm{h}} \\ \Leftrightarrow \quad & \mathrm{L_h}(\mathbf{Q}_{\mathrm{h}}^{\infty}) - \mathrm{L_h}(\mathbf{Q}_{\mathrm{err,h}}^{k}) = \mathbf{B}_{\mathrm{h}} - \mathbf{R}_{\mathrm{h}}^{k} \\ \Leftrightarrow \quad & \mathrm{L_h}(\mathbf{Q}_{\mathrm{err,h}}^{k}) = \mathbf{R}_{\mathrm{h}}^{k} \,. \end{aligned} \tag{5.17}$$

A linear multi-grid algorithm will transfer the residual to a coarser grid and then iterate the error-equation on the coarser grid. With the new approximation for the error on the coarse grid, the values for $\mathbf{Q}_{\mathrm{h}}^{k}$ can be corrected.

5.2.1 Transfer Operators

Consider two different grids, a fine grid and a coarser one. Operators shall exist to transfer values from one grid to another. $\mathrm{T}_{\mathrm{r,h\to H}}$ denotes an operator to transfer the residuals from fine to coarse. $\mathrm{T}_{\mathrm{H\to h}}$ represents an operator to transfer values for the error from coarse to fine. The nature of the residuals $\mathbf{R}_{\mathrm{h}}^{k}$ and the error $\mathbf{Q}_{\mathrm{err,h}}^{k}$ or $\mathbf{Q}_{\mathrm{err,H}}^{k}$ is significantly different. For a finite-volume discretization $\mathbf{Q}_{\mathrm{h}}^{k}$ and $\mathbf{Q}_{\mathrm{err,h}}^{k}$ would be volume-specific properties, whereas the residuals would be volume-integrated values. This has to be taken into account for the transfer operators. Figure 5.3 shows two different possibilities of constructing coarse levels. The nested version will create coarse grid control-volumes by simply adding fine grid volumes. This is quite often called agglomeration and its use for unstructured grids will be described in detail in paragraph 5.4. When this approach is

used, every fine grid node, or control volume, is uniquely linked to a coarse node. For this simple example, however, the non-nested version is more suited, since it allows a better construction of the coarse to fine operator. A possible approach for the operators $\mathrm{T}_{\mathrm{r,h\to H}}$ and $\mathrm{T}_{\mathrm{H\to h}}$ could be the following:

$$\mathrm{T}_{\mathrm{r,h\to H}} = \begin{pmatrix} 1 & \frac{1}{2} & 0 & \dots & & & & & & \\ 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & \dots & & & & \\ & \dots & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & \dots & & \\ & & & & & \ddots & & & & \\ & & & \dots & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & \dots \\ & & & & & \dots & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 \\ & & & & & & & \dots & 0 & \frac{1}{2} & 1 \end{pmatrix}, \quad \mathbb{R}^{N_f} \to \mathbb{R}^{N_c} \tag{5.18}$$

$$\mathrm{T}_{\mathrm{H\to h}} = \begin{pmatrix} 1 & 0 & \dots & & & \\ \frac{1}{2} & \frac{1}{2} & 0 & \dots & & \\ 0 & 1 & 0 & \dots & & \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & \dots & \\ & & & \ddots & & \\ & \dots & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ & & \dots & 0 & 1 & 0 \\ & & \dots & 0 & \frac{1}{2} & \frac{1}{2} \\ & & & \dots & 0 & 1 \end{pmatrix}, \quad \mathbb{R}^{N_c} \to \mathbb{R}^{N_f} \tag{5.19}$$

N_f and N_c represent the number of fine and coarse nodes.

5.2.2 Ideal Two Grid Cycle

The following iterative procedure is called an ideal two grid cycle:

1. An initial approximation $\mathbf{Q}_{\mathrm{h}}^{k}$ is computed on the fine grid. This means the short-wave errors are smoothed and hence this will often be called pre-smoothing. A Jacobi iteration is very often used for these smoothing steps.

2. The fine grid residual is transferred to the coarse grid. Multi-grid terminology calls this *restriction*.
$$\mathbf{R}_{\mathrm{H}} = \mathrm{T}_{\mathrm{r,h\to H}}(\mathbf{R}_{\mathrm{h}}) \tag{5.20}$$

3. The equation for the error
$$\mathrm{L}_{\mathrm{H}}(\mathbf{Q}_{\mathrm{err,H}}) = \mathbf{R}_{\mathrm{H}} \tag{5.21}$$
is solved on the coarse grid. Please note that the operators on the fine grid L_{h} and on the coarse grid L_{H} are different.

4. On the fine grid $\mathbf{Q}_{\mathrm{h}}^{k}$ can now be corrected, using the following formula.
$$\mathbf{Q}_{\mathrm{h}}^{k+1} = \mathbf{Q}_{\mathrm{h}}^{k} + \mathrm{T}_{\mathrm{H\to h}}(\mathbf{Q}_{\mathrm{err,H}}) \tag{5.22}$$
Transfer from a coarse to a fine grid is called *prolongation*.

5. Possible short-wave errors which might have been introduced by the transfer operator $\mathrm{T}_{\mathrm{H}\to\mathrm{h}}$ can now be eliminated by running a few iteration on the fine grid.

The ideal two grid cycle can be sketched as follows:

$$\begin{array}{ccccc} \mathbf{Q}_{\mathrm{h}}^{k=1}\to\mathbf{Q}_{\mathrm{h}}^{2}\to\ldots\to\mathbf{Q}_{\mathrm{h}}^{N_{\mathrm{pre}}} & & & & \mathbf{Q}_{\mathrm{h}}^{k=N_{\mathrm{pre}+1}}\to\mathbf{Q}_{\mathrm{h}}^{N_{\mathrm{pre}+2}}\to\ldots\to\mathbf{Q}_{\mathrm{h}}^{N_{\mathrm{pre}}+N_{\mathrm{post}}} \\ & \searrow & & \nearrow & \\ & & \mathbf{Q}_{\mathrm{H}}^{K=1}\to\mathbf{Q}_{\mathrm{H}}^{2}\to\ldots\to\mathbf{Q}_{\mathrm{H}}^{\infty} & & \end{array}$$

Please note that capital indices (H,K) denote values on the coarse level, whereas lowercase indices are used for the fine level. A detailed study of Jacobi or Gauss-Seidel iterations as smoothing operators can be found in [21].

5.2.3 The Standard V-Cycle

The process, which has been described in the previous paragraph, requires a solution of (5.21) within the desired convergence limit. This is still fairly expansive in terms of computational time. In fact the complexity will still be quadratic ($N_{\mathrm{comp}} \sim {N_{\mathrm{nds}}}^2$). To improve this, for a given mesh, a sequence of grids will be created. The goal is to have a coarsest grid with a constant number of nodes, no matter how many nodes there are on the finest grid. Figure 5.4 illustrates how the solution process will look like. It is fairly obvious, why this is called a V-cycle. The computational complexity of this algorithm becomes:

$$N_{\mathrm{comp}} \sim N_{\mathrm{nds}} \cdot \log(N_{\mathrm{nds}}) \tag{5.23}$$

with N_{comp} as the total amount of computations needed and N_{nds} as the number of nodes on the finest grid. A linear complexity can be reached with the so called full multi-grid algorithm. Here the iterative process is started on the coarsest grid. The solution, which will be obtained, is then used as initial guess for the next finer grid. A solution on this grid is computed using multi-grid. This is consequently done until a solution on the finest grid has been computed. Please note that the complexity issues mentioned in this paragraph are only valid for the linear multi-grid. For the non-linear multi-grid, as it will be described in the next paragraph, the behavior is not as analyzable as it is here.

5.3 The FAS method

The above described linear multi-grid algorithm can not be used to accelerate the convergence of a solution process for non-linear equations. Consider the following non-linear equation:

$$\Phi(\mathbf{q}) = 0 \tag{5.24}$$

where Φ is any non-linear operator. A discretized form can be written similarly to (5.12):

$$\Phi_{\mathrm{h}}(\mathbf{Q}_{\mathrm{h}}) = 0 \tag{5.25}$$

$$\Phi_{\mathrm{h}}(\mathbf{Q}_{\mathrm{h}}^{k}) + \mathbf{R}_{\mathrm{h}}^{k} = 0\,. \tag{5.26}$$

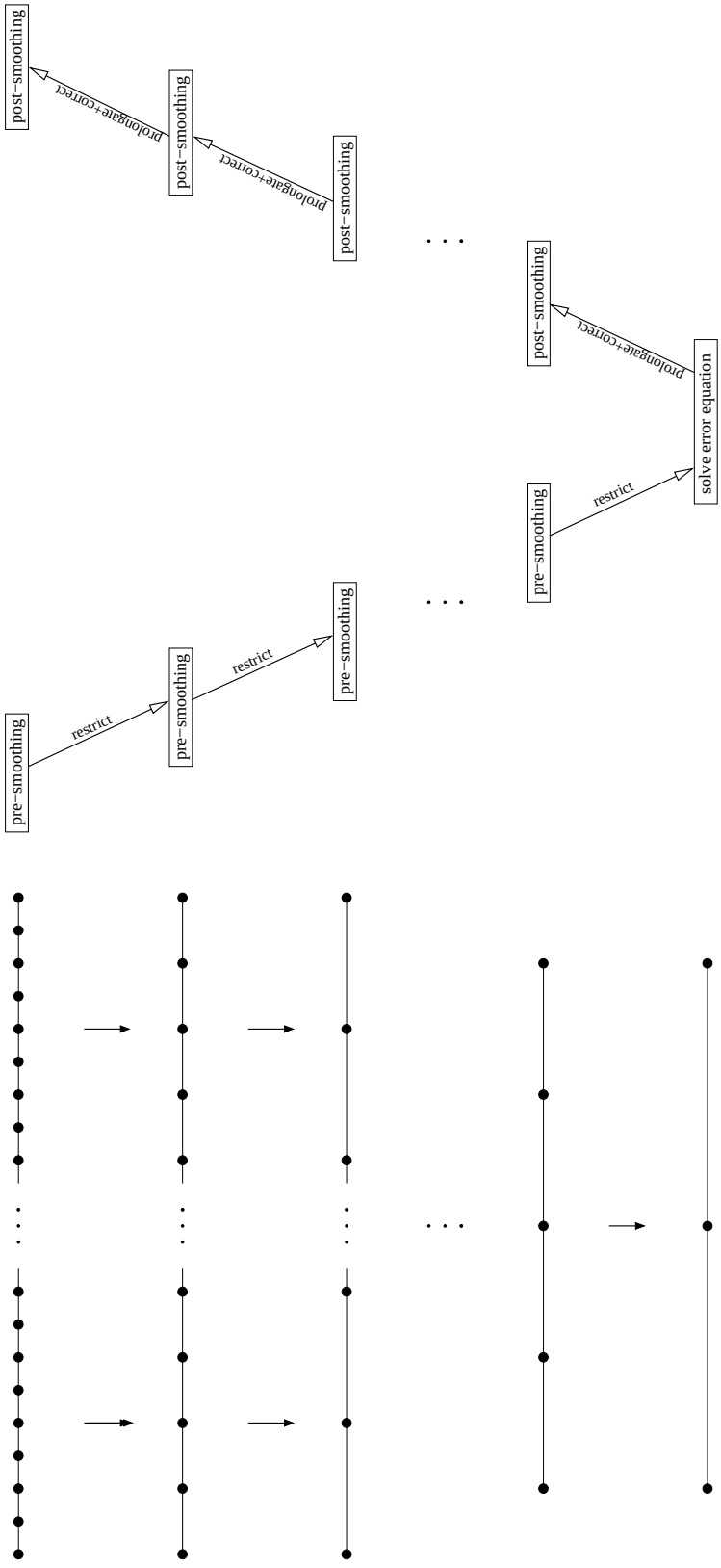

Figure 5.4: A V-cycle.

Unfortunately, due to its non-linearity, the operator Φ does not allow a superposition like L does (see (5.16)). Thus the complete set of equations has to be transferred to the coarse grid, instead of solving an equation for the error only. Hence this type of multi-grid is called *full approximation scheme* or sometimes *full approximation storage*. Here the FAS method shall be briefly explained. This is done, using a two grid cycle, similar to the example for the linear scheme. As for the linear multi-grid method, inter-grid operators are needed. $\mathrm{T}_{\mathrm{h}\to\mathrm{H}}$ is used to restrict the unknowns and $\mathrm{T}_{\mathrm{r,h}\to\mathrm{H}}$ the residuals. The prolongation of the correction from the coarse to the fine grid is written as $\mathrm{T}_{\mathrm{H}\to\mathrm{h}}$. The equation which will be used for the coarse grid iterations is the full non-linear equation, but discretized on the coarse grid:

$$\Phi_{\mathrm{H}}(\mathbf{Q}_{\mathrm{H}}) \;=\; \mathbf{f}_{\mathrm{H}}\,. \tag{5.27}$$

To compute the right hand side for the coarse grid equation, (5.26) shall be transferred to the coarse grid:

$$\Phi_{\mathrm{H}}(\mathrm{T}_{\mathrm{h}\to\mathrm{H}}(\mathbf{Q}_{\mathrm{h}}^{k})) \;+\; \mathrm{T}_{\mathrm{r,h}\to\mathrm{H}}(\mathbf{R}_{\mathrm{h}}^{k}) \;=\; \mathbf{f}_{\mathrm{H}}\,. \tag{5.28}$$

Please note that the term $\mathbf{f}_{\mathrm{H}}$ hinders the coarse grid iteration in moving away too far from the fine grid values. If (5.25) has been exactly fulfilled on the fine grid, $\mathbf{R}_{\mathrm{h}}^{k}$ will be zero and hence the discrete equation on the coarse grid becomes:

$$\begin{aligned}
& \mathbf{R}_{\mathrm{h}}^{k} && = 0 \\
\Rightarrow\quad & \mathrm{T}_{\mathrm{r,h}\to\mathrm{H}}(\mathbf{R}_{\mathrm{h}}^{k}) && = 0 \\
\Rightarrow\quad & \mathbf{f}_{H} && = \Phi_{\mathrm{H}}(\mathrm{T}_{\mathrm{h}\to\mathrm{H}}(\mathbf{Q}_{\mathrm{h}}^{k})) \\
\Rightarrow\quad & \Phi_{\mathrm{H}}(\mathbf{Q}_{\mathrm{H}}) && = \Phi_{\mathrm{H}}(\mathrm{T}_{\mathrm{h}\to\mathrm{H}}(\mathbf{Q}_{\mathrm{h}}^{k}))\,.
\end{aligned} \tag{5.29}$$

Since $\mathrm{T}_{\mathrm{h}\to\mathrm{H}}(\mathbf{Q}_{\mathrm{h}}^{k})$ is used as initial guess for the coarse grid equation, no change will take place at all in this case. Normally the fine grid values will be corrected using the following formulation:

$$\mathbf{Q}_{\mathrm{h}}^{k+1} \;=\; \mathbf{Q}_{\mathrm{h}}^{k} \;+\; \mathrm{T}_{\mathrm{H}\to\mathrm{h}}\left(\mathbf{Q}_{\mathrm{H}} \;-\; \mathrm{T}_{\mathrm{h}\to\mathrm{H}}(\mathbf{Q}_{\mathrm{h}}^{k})\right)\,. \tag{5.30}$$

The ideal two grid FAS algorithm is very similar to the linear version:

1. pre-smoothing
2. restriction of unknowns and residuals
3. solution of the coarse grid equation
4. prolongation and correction
5. post-smoothing

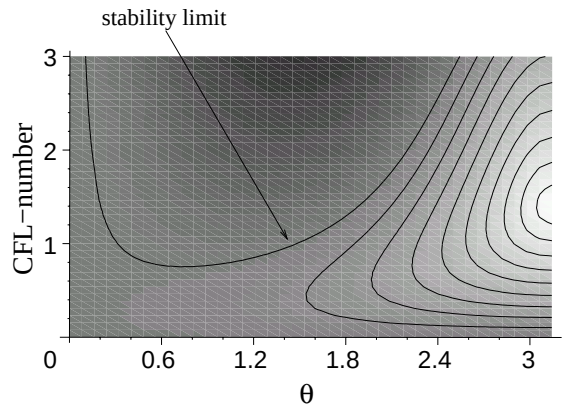

Figure 5.5: Amplification factor ($\beta = \frac{1}{2}$).

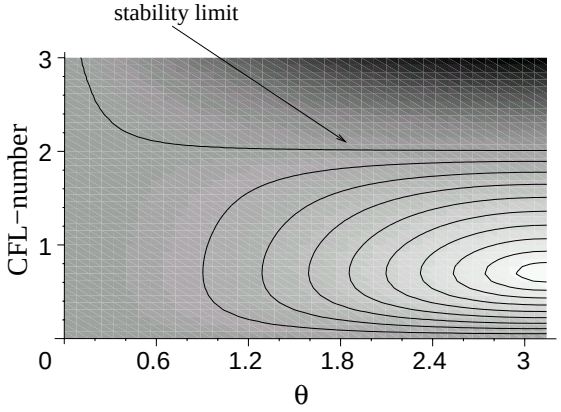

Figure 5.6: Amplification factor ($\beta = 1$).

5.3.1 Runge-Kutta Scheme for Smoothing

The FAS algorithm allows to treat non linear equations. On every level the exact equation, including all boundary conditions, has to be modeled. In the presented FAS implementation an explicit Runge-Kutta is used for smoothing, as well as for solving the equation on the coarsest level. For linear multi-grid methods a Jacobi or Gauss-Seidel relaxation is often used for smoothing. Unfortunately, this requires a point implicit formulation of the discretization. To allow the method to be easily used with different discretizations, a fully explicit Runge-Kutta method has been chosen. The main problem when using Runge-Kutta as a smoother, is that the smoothing properties are dependent on the number of sub-steps, the coefficients used, the Courant number and the discretization itself. A Runge-Kutta scheme, using the same coefficients, will have completely different damping properties if used for an upwind or a central discretization. In contrast to the example equation (5.11), many problems related to fluid mechanics are convection dominated. To illustrate the difficulties, while predicting the damping behavior of a Runge-Kutta scheme as smoother, the following model-equation shall be analyzed.

$$\frac{\partial \phi}{\partial t} + u \frac{\partial \phi}{\partial x} = 0 \quad , \quad u > 0 \tag{5.31}$$

Let u be a constant convection speed in the whole domain. (5.31) shall be discretized, using a mixed upwind/central discretization on an equidistant Cartesian mesh (edge-length h):

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} + u \left(\beta \frac{\phi_i^n - \phi_{i-1}^n}{h} + (1-\beta) \frac{\phi_{i+1}^n - \phi_{i-1}^n}{2\,h} \right) = 0\,, \tag{5.32}$$

with β as parameter to switch between upwind or central discretization ($\beta = 1$ represents a full upwind and $\beta = 0$ a central scheme). Time integration is done by a three step Runge Kutta scheme with the coefficients $\alpha_1 = \frac{1}{4}, \alpha_2 = \frac{1}{2}, \alpha_3 = 1$.
Figures 5.5 and 5.6 show the damping properties of this scheme. A von Neumann analysis gives an amplification factor of this scheme as a function of the time step Δt and the

wave-angle $\Theta \in [0..\Pi]$. $\Theta = \Pi$ represents the shortest and $\Theta = 0$ the longest resolvable wavelength. Dark regions in figures 5.5 and 5.6 represent a high amplification. Isolines are drawn for amplification values of $(0.1, 0.2, 0.3, ..., 1.0)$. Please note that values > 1 indicate an unstable scheme.
The figures clearly show the difficulties related to this type of smoother. To obtain maximal efficiency of the overall algorithm, the smoother should be used in the area of maximal damping for high frequency errors. Unfortunately it is close to impossible to accurately predict the real CFL-number for a multi-dimensional and non-linear system of equations, as for example the Euler or Navier-Stokes equations. The main advantage is, that the same discretization (i.e. fluxes, source-terms etc.) can be used for time-explicit unsteady simulations and FAS steady state computations. Therefore this type of smoothers has been chosen, despite its disadvantages. Simulations, which have been made using this type of multi-grid proved to be about as efficient, as a more classical implicit and linear multi-grid combination.
As it is noted in [21], the smoother does not have to be convergent. It has to be efficient for high frequency disturbances. In figure 5.5 it can be seen that best damping for high frequencies can be obtained by an unstable setting. Unfortunately, within the scope of this work, it has not been possible to apply a theoretically unstable Runge Kutta scheme with good smoothing properties for high frequencies. At least it has not been possible in a reliable way. Hence a more detailed investigation of the convergence and smoothing properties might pay off. Even if linear model equations shall be investigated, it still is an extremely difficult task on unstructured grids. The definition of a CFL-number on a particular node of such a grid is difficult, due to the different lengths and directions of the adjacent edges.

5.4 Construction of Mesh Sequences

The agglomeration technique is used to build a set of coarse finite volume grids. A coarser level can be obtained by clustering neighboring volumes. Each cluster of volumes will form a control volume of the coarsened mesh. Used recursively, this technique produces a sequence of grids with nested control volumes. Figures 5.7 and 5.8 show the principle of this agglomeration algorithm applied to a mixed element sample mesh. The most used technique to agglomerate control volumes is a greedy-like algorithm. But for 3D problems, Mavriplis[20] proposes a priority list to initialize a pure greedy technique. In [22] a coloring edge-based method is found. These variants show an interest on the quality of the agglomerated cells. The basic agglomeration, as it has been described above, can be improved by using a local optimization criterion. It can be observed that the regularity of the agglomerated levels influences the convergence rate. So far the break-even between time spent on optimizing an agglomeration and improvement of the convergence speed, could not be reached. This is, however, mainly due to non-linear complexities in the optimization algorithm. Approaches to overcome those exist, but they could not be implemented in the scope of this work.

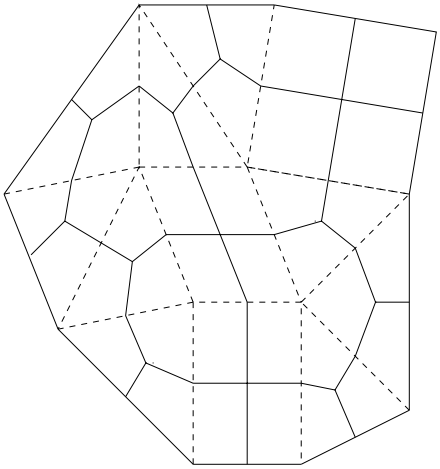

Figure 5.7: Elements and associated control volumes.

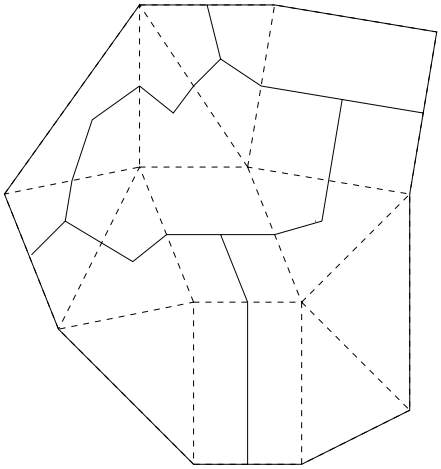

Figure 5.8: Elements and agglomerated control volumes.

5.4.1 Optimizing the Shape of Agglomerated Cells

Agglomerated grids can be optimized, using a strategy which is based on the detection of so called 'horseshoe' cells. These cells are cells, that are not purely convex. To avoid cells of this type, the following algorithm can be applied. Before any optimization will be used, an agglomeration has to be computed, using the standard algorithm. Every cell, on every level of the grid sequence, can be viewed as a closed polygon with a set of normal vectors. An ellipse shall now be constructed with a shape similar to the original cell (see figures 5.9 and 5.10). Once this shape has been constructed, the area ratio $\frac{A_{\text{ellipse}}}{A_{\text{cell}}}$ is a measure for the presence of horseshoes. To construct the replacing ellipse, values for $r_{\min}$ and $r_{\max}$ are needed. They are defined as follows:

$$r(\alpha) = \sum_{i=1}^{N_{\text{seg}}} \left| \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} \cdot \mathbf{n}_i \right| ,$$

$$r_{\min} = \min(r(\alpha)) \,,\, \alpha \in [0, \Pi] \quad , \quad r_{\max} = \max(r(\alpha)) \,,\, \alpha \in [0, \Pi] \,. \tag{5.33}$$

Where N_{seg} is the polygon's number of segments and $\mathbf{n}_i$ is the normal vector of segment i. Please note that 'horseshoe' like cells appear to be bigger than they really are, when using this method to evaluate $r_{\max}$. Evaluating (5.33) is fairly time consuming. The extrema of a continuous function have to be computed. After these critical cells have been spotted, they can be improved by exchanging contributing fine grid cells between two adjacent coarse grid control volumes. Besides this shape criterion other criteria can be employed. The ratio $\frac{r_{\max}}{r_{\min}}$ can serve as an estimate for anisotropy. Based on an already agglomerated level the optimizer tries to minimize a function, which can consist of several evaluation criteria. How this optimization has an impact on the convergence rate can be found in chapter 8.

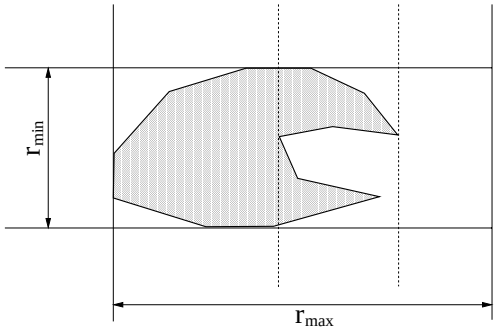

Figure 5.9: Cell to evaluate.

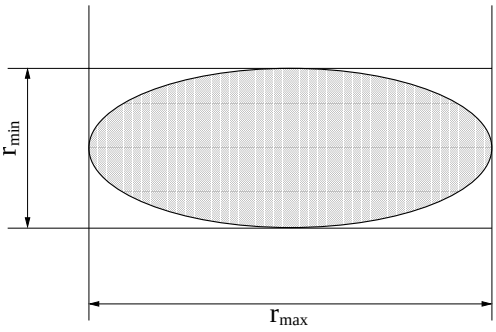

Figure 5.10: Replacing ellipse.

5.5 Higher Order Spatial Discretization

Normally a second order flux computation is performed using MUSCL interpolation. This technique is based on a reconstruction of the state on the cell interfaces, using linear extrapolation. The extrapolation is performed using the gradients of the variables, which are computed beforehand, using a least square algorithm or the divergence theorem (see B). On the coarse levels, created by the above described agglomeration technique, the position of the nodes and the control volume borders become irregular. This makes a second order discretization using MUSCL extrapolation very difficult. Experiences show that such a discretization is very critical in terms of stability for the explicit time stepping. Therefore it does not seem to be reasonable to apply a higher than first order approaches on the coarse grids.
In order to achive an accurate discretization on the fine level, the defect correction [23] method has been used. The basic idea of this method is to split up the iteration into two levels. A higher order discretization is realized by a source term, that influences a first order scheme, which is used to advance the solution. The following discrete equation is to be solved:

$$\Phi_{\mathrm{tgt}}(\mathbf{Q}) \;=\; 0\,. \tag{5.34}$$

Φ_{tgt} denotes the target (higher order) discretization. Any intermediate state $\mathbf{Q}^k$ will be advanced by a driving (typically first order) discretization (denoted by Φ_{drv}):

$$\Phi_{\mathrm{drv}}(\mathbf{Q}^{k+1,\nu=0..N_{\mathrm{iter}}}) \;=\; \Phi_{\mathrm{tgt}}(\mathbf{Q}^k) \;-\; \Phi_{\mathrm{drv}}(\mathbf{Q}^k)\,. \tag{5.35}$$

The number and type of the iterations in ν can vary. It is not necessary to fully converge the driving iteration. Good experiences, however, have been made with a single FAS V-cycle as driving iteration. If the series converges (in k),

$$\begin{aligned} \Phi_{\mathrm{drv}}(\mathbf{Q}^{k+1}) \;&=\; \Phi_{\mathrm{drv}}(\mathbf{Q}^k) \\ \Rightarrow \Phi_{\mathrm{tgt}}(\mathbf{Q}^k) \;&=\; 0\,. \end{aligned} \tag{5.36}$$

will be fulfilled. This technique allows to compute second order discretizations, without having to discretize higher than first order on the agglomerated grids.

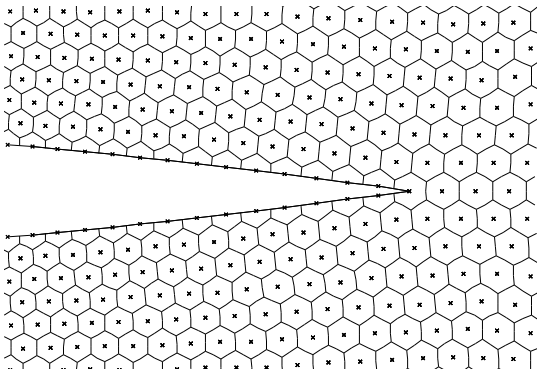

Figure 5.11: Fine grid control volumes.

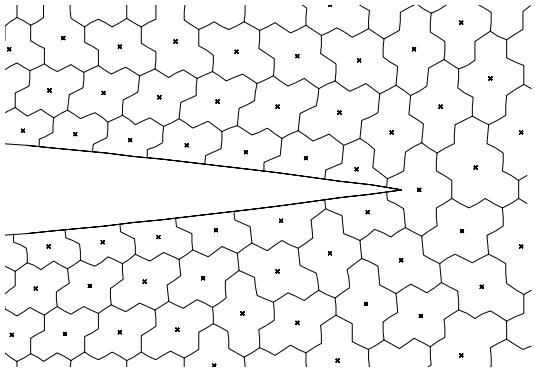

Figure 5.12: 1st agglomerated level.

5.6 Boundary Conditions

The FAS multigrid algorithm needs a correct expression of the boundary conditions on every grid level. As explained in 5.3 a partially converged solution is obtained on each grid. On every level the same set of equations is modeled. The only difference can be found in the position of the boundary nodes. Only on the finest mesh the nodes are on the boundary itself. This makes the implementation of nodal formulations for boundary conditions easy. For the agglomerated levels this is not the case anymore. The node position can no longer be assumed to be on the boundary. Figures 5.11 and 5.12 illustrate this.
Furthermore, geometric properties (e.g. surface normal vectors) are difficult, if not impossible, to reconstruct on the coarse levels. To still imply the boundary conditions on the coarsened levels, they are included in the right hand side of the defect correction scheme. If BC defines an operator representing the nodal boundary conditions (e.g. a point-wise non-slip condition), the right hand side can be altered:

$$\Phi_{\mathrm{drv}}(\mathbf{Q}^{k+1,\nu=0..N_{\mathrm{iter}}}) = \mathrm{BC}\left(\mathbf{Q}^k + \Phi_{\mathrm{tgt}}(\mathbf{Q}^k)\right) - \mathbf{Q}^k - \Phi_{\mathrm{drv}}(\mathbf{Q}^k)\,. \tag{5.37}$$

Using this formulation it is possible to enforce the boundary conditions on the coarse levels. This can be done, without having to actually formulate them on an agglomerated grid, which would be very difficult. Some types of boundary conditions, however, cause troubles when used with this formulation. The non-slip condition for a Navier-Stokes computation is a good example. A way to overcome these problems is to modify (5.37) in the following way:

$$\Phi_{\mathrm{drv}}(\mathbf{Q}^{k+1,\nu=0..N_{\mathrm{iter}}}) = \omega\left(\mathrm{BC}\left(\mathbf{Q}^k + \Phi_{\mathrm{tgt}}(\mathbf{Q}^k)\right) - \mathbf{Q}^k\right) - \Phi_{\mathrm{drv}}(\mathbf{Q}^k)\,. \tag{5.38}$$

ω represents an adjustable factor to slow down the iteration advance and it has to be chosen between $\omega = 0$ and $\omega = 1$. For practical simulations a good strategy has been to start with a fairly low value of ω and then gradually try to increase it towards $\omega = 1$.

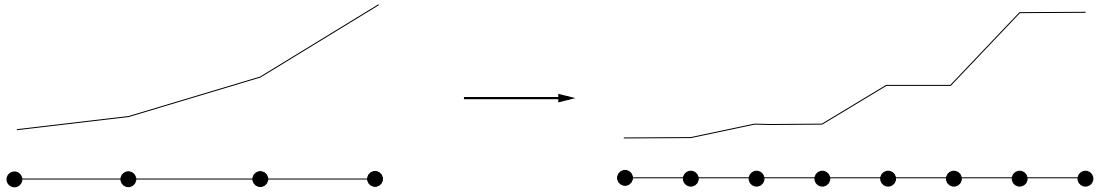

Figure 5.13: Prolongation by canonical injection.

5.7 Inter-Grid Operators

Transfer operators are usually considered to be a problematic part of the volume agglomeration technique. The restriction for the residual is simply a summation of all fine grid cells which contribute to one agglomerated cell. To restrict the variable vector an averaging over all contributing cells is employed. Please note that this is done, using the control volume sizes as weighting factors. As prolongation operator an extremely simple canonical injection has been used (see also [19]). The values from coarse cells are directly transferred to all fine grid cells which belong to the coarse one. By doing so a piecewise constant function will be created on the fine grid (see figure injection). This can lead to severe problems. For a gas-dynamical problem for instance, these steps could be seen as small shock waves. Depending on the number of post smoothing steps, this might lead to an amplification of those disturbances, which are caused by the prolongation operator. One way to overcome this amplification problem is to smooth the prolongated values. A strategy, similar to what is used for residual-smoothing (see 3.3.2) has been used and proved to be very efficient. Another strategy, that has been tried, is to linearly reconstruct the fine grid values, by using the coarse grid gradients. This, however, did not lead to significant improvements and therefore the simpler smoothing strategy has been given preference.

5.8 Impact of the Agglomeration Quality on the Convergence

The following numerical experiment compares the convergence rate of an optimized and a non-optimized agglomeration. As computational example serves a subsonic ($M = 0.5$) flow around a NACA0012 airfoil. The angle of attack is 2 degrees. Target discretization is a second order Roe-scheme and as driver a first order Roe-scheme is used. Figure 5.14 shows a comparison of the coarsest level for the standard and optimized agglomeration. For this test case a small improvement in terms of convergence behavior can be observed (see figure 5.16). $(\Delta\rho)_{\text{expl}}$ is the change in density of an explicit control-step, which has been executed after every V-cycle. It has been normed by the maximum.
For anisotropic regions the coarse grid quality will become a crucial requirement to obtain good convergence improvements (see also [24]). The present library, however, does not yet allow the usage of anisotropic grids for multi grid computations. Another problem is

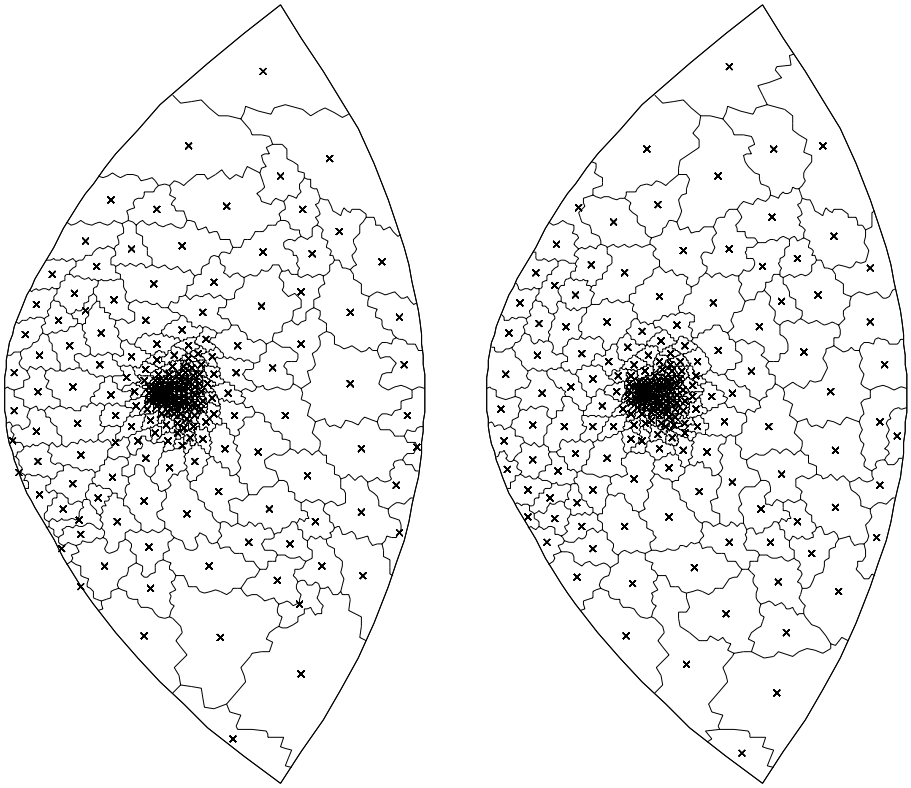

Figure 5.14: Standard(left) and optimized(right) agglomeration.

introduced by quadrangular regions in hybrid grids. The natural orientation and node-numbering yields to sometimes unwanted effects. Very often the main axes of bilinear elements are turned by 45 degrees when going to a coarser grid level. This leads to rhombic elements, which especially in anisotropic layers are most unwanted. Besides the above described shape optimization there is another approach to overcome this problem. The so-called semi-coarsening [24], tries to coarsen an anisotropic grid only in one distinct direction.

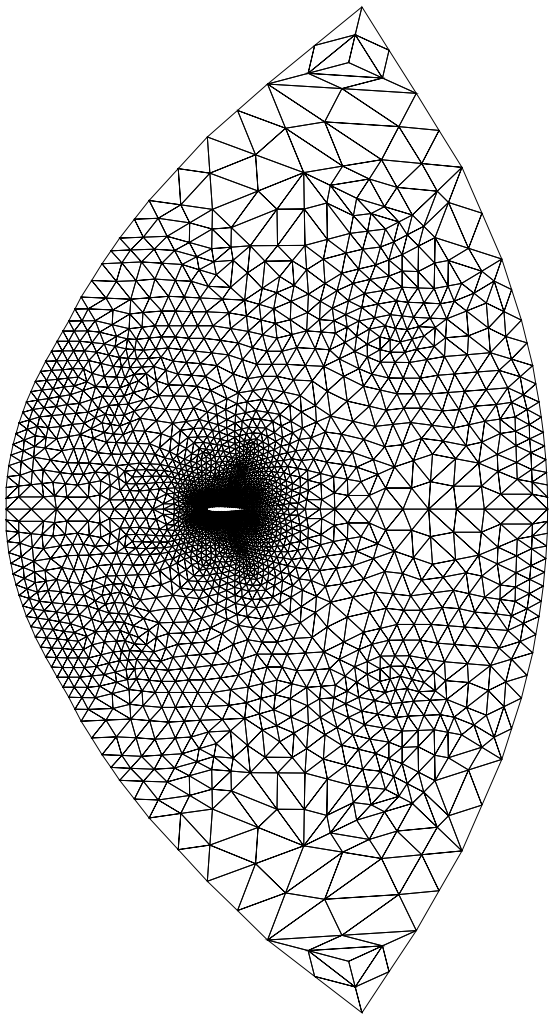

Figure 5.15: Fine grid.

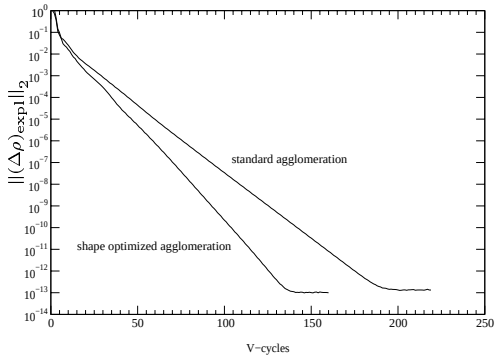

Figure 5.16: Convergence comparison.