European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2006
P. Wesseling, E. Oñate and J. Périaux (Eds)
© TU Delft, The Netherlands, 2006

ANALYSIS OF A HYBRID P-MULTIGRID METHOD FOR THE DISCONTINUOUS GALERKIN DISCRETISATION OF THE EULER EQUATIONS

Koen Hillewaert[♠], Jean-François Remacle[†], Nicolas Cheveaugeon[‡], Paul-Emile Bernard[*] and Philippe Geuzaine[♣]

[♠,♣] CENAERO, CFD-MP group
Av. J. Mermoz 30, B6041 Gosselies, Belgium
http://www.cenaero.be
[♠] koen.hillewaert@cenaero.be
[♣] philippe.geuzaine@cenaero.be

[†,‡] UCL, Unité du Génie Civil et Environnemental
Bâtiment Vinci, Place du Levant 1, B1348 Louvain-La-Neuve, Belgium
http://www.gce.ucl.ac.be
[†] remacle@gce.ucl.ac.be
[‡] cheveaugeon@gce.ucl.ac.be

[*] UCL, Unité de Mécanique Appliquée
Bâtiment Euler, Av. G. Lemaître 4, B1348 Louvain-La-Neuve, Belgium
http://www.mema.ucl.ac.be
[*] bernard@mema.ucl.ac.be

Key words: Discontinuous Galerkin Finite Element Method, p-Multigrid, h-Multigrid, Implicit Solvers

Abstract.

An elegant yet practical framework for the application of p- and h-Multigrid for DGFEM is first presented and analysed theoretically.

After that a hybrid implicit-explicit p-Multigrid iteration strategy for the discretisation of the steady Euler equations with the discontinuous Galerkin finite element method is presented and investigated experimentally. The implicit strategy consists of an inexact damped Newton iteration, using an ILU(0)-preconditioned matrix-free GMRES method for the solution of the linear system. The size of the ILU preconditioner grows very fast with interpolation order. As a result the method becomes impractical already for moderate orders of interpolation. Therefore it is embedded in a FAS p-Multigrid iteration scheme. In this framework, the implicit solver is used only on the lowest order interpolation space, while on the higher order levels a Runge-Kutta local timestepping method is used. The fast convergence of the solution on the lowest order levels speeds up the convergence of the mainly explicit multilevel iterations considerably with respect to a purely explicit p-Multigrid solver.

1 INTRODUCTION

The Discontinuous Galerkin Finite Element Method (DGFEM) has recently become the topic of intense research for the computation of hyperbolic and elliptic problems. This interest is due to the combination of an arbitrary order of accuracy with data and algorithmic locality. Since the interpolation functions are defined independently in each element the mass matrix has a block-diagonal structure and may be inverted directly. The method can be easily parallelized because DGFEM stencils do not grow in size with increasing order. The interpolation functions are not restrained by continuity requirements and can hence be chosen freely, allowing easy implementation of e.g. spectral interpolation bases. Another consequence is the inherent capability of the method to handle h- and p-adaptation.

In Computational Fluid Dynamics the method has become fashionable for the computation of unsteady flows, especially when accurate solutions are needed, such as direct or large eddy simulation of turbulence and the computation of shock propagation. It has however not yet experienced a similar interest for the computation of steady flows. On one hand the accuracy requirements are not seen to be as critical as for unsteady computations. On the other hand DGFEM still has some evident drawbacks, mainly in terms of computational efficiency. First and foremost an efficient iteration method to attain the steady state is still lacking. Moreover the computation of the residual and such are still quite computationally intensive tasks. Another problem concerns the stabilisation of the method in case of solution discontinuities.

Recent papers of Fidkowski et al.[5] and Hartmann[6] have demonstrated that in terms of CPU time for a given precision it is better to increase DGFEM interpolation order rather than mesh resolution, at least for smooth solutions. Hence it is possible that higher-order DGFEM method may in time grow to be more efficient than a state-of-the-art finite volume solver for steady computations, especially if combined with h- and p- adaptation. Another advantage is that accurate results can be obtained with — by today's standards — very coarse meshes, which relaxes the requirements for the mesh generators (which often are not parallellized and are hence limited in mesh size) and diminishes the memory overhead related to storage of mesh topology.

In this paper we investigate an efficient iteration strategy that embeds an implicit solver into a p-Multigrid framework, exploiting its efficiency while limiting the memory footprint and algorithmical cost by using explicit solvers on the finest levels.

The paper is organized as follows. First DGFEM is reviewed briefly. After that we discuss and analyse a general framework for p- and h-Multigrid for DGFEM. In the next section we discuss a number of p-Multigrid strategies. In the last section we discuss and compare several iteration strategies for DGFEM to the p-Multigrid method.

2 THE DISCONTINUOUS GALERKIN FINITE ELEMENT METHOD

Suppose we want to approximate the solution u to the following unsteady convection problem on the domain $\mathcal{D}$:

$$\mathcal{L}(u) = f \tag{1}$$

where we have defined the residual operator $\mathcal{L}()$ as:

$$\mathcal{L}(\alpha) = \nabla \cdot \vec{\mathcal{F}}(\alpha) \tag{2}$$

The *Discontinuous Galerkin Finite Element Method* (DGFEM) approximates the solution u in the broken or discontinuous interpolation space $\mathcal{U}^p$ that is spanned by shape functions ϕ_i^p. Each of these functions ϕ_i^p is defined on one element V only of a tesselation $\mathcal{V}$ of $\mathcal{D}$. The approximate solution u^p is as such discontinuous across element boundaries.

$$u^p = \sum_i \mathbf{u}_i^p \ \phi_i^p \ , \ \phi_i^p \in \mathcal{U}^p \tag{3}$$

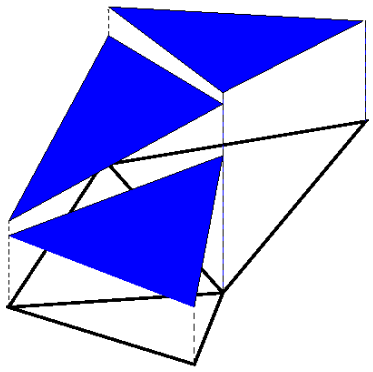


Figure 1: DGFEM interpolation space

The expansion coefficients $\mathbf{u}_i^p$ are found by requiring that the approximate solution u^p satisfies the following weak formulation of the equations (1):

$$(\psi^p, \mathcal{L}(u^p)) = (\psi^p, f) \ , \ \forall \psi^p \in \mathcal{U}^p \tag{4}$$

where the inner product in the broken space $\mathcal{U}^p$ has been defined as:

$$(\alpha^p, \beta^p) = \int_{\mathcal{D}} \alpha^p \ \beta^p \ dV \ , \ \forall \alpha^p, \beta^p \in \mathcal{U}^p \tag{5}$$

Considering that any $\psi^p \in \mathcal{U}^p$ is a linear combination of the shape functions ϕ_i^p, and that any shape function ϕ_i^p is supported and continuous on one element V of $\mathcal{V}$ only we can rewrite (4) as:

$$\begin{gathered} \int_{\mathcal{D}} \phi_j^p \ \left(\nabla \cdot \vec{\mathcal{F}}(u^p)\right) \ dV = 0 \ , \ \forall \phi_j^p \in \mathcal{U}^p \\ \oint_{\partial V} \phi_j^p \left(\vec{\mathcal{F}}(u^p) \cdot \vec{n}\right) dS - \int_V \nabla \phi_j^p \cdot \vec{\mathcal{F}}(u^p) \ dV = 0 \end{gathered} \tag{6}$$

Since the data are discontinuous, the boundary flux $\vec{\mathcal{F}}(u^p) \cdot \vec{n}$ is not defined. We replace it by a numerical flux function $\mathcal{H}$ that depends on the solution from both sides:

$$\oint_{\partial V} \phi_j^p \left(\mathcal{H}(u_+^p, u_-^p, \vec{n})\right) \ dS - \int_V \nabla \phi_j^p \cdot \vec{\mathcal{F}}(u^p) \, dV = 0 \tag{7}$$

3 MULTIGRID FOR DGFEM

h-Multigrid methods for DGFEM have been presented and analysed by Bastian et al.[2] and van der Vegt et al.;[11] p-Multigrid methods by Helenbrook et al.,[7] Fidkowski et al.[5] and Luo et al.[9]

In this section we present and analyse an elegant yet practical framework for both h- and p-Multigrid applied to DGFEM. Most of this section is an adaptation of the framework for analysis that has been summarised in chapters 2 and 6 in the classical textbook by Wesseling.[12]

Consider now another interpolation space $\mathcal{U}^q$ which is coarser then $\mathcal{U}^p$ in some way. In case of p-Multigrid, the space $\mathcal{U}^q$ is based on the same tesselation $\mathcal{V}$ as $\mathcal{U}^q$, while the order of interpolation is lower; for h-Multigrid the interpolation is of the same order, but is based on a different, less resolved tesselation.

3.1 The two-level FAS cycle

The classical two-level FAS cycle[12] may then be redefined for a non-linear finite element method as follows:

1. Perform a number of iterations on level p, leading to solution $u^{p\prime}$
2. *Restrict* the current solution $u^{p\prime}$ to $u^{q\prime}$ in $\mathcal{U}^q$ by a suitable operator $\mathcal{T}^{qp}$:
$$u^{q\prime} = \mathcal{T}^{qp}\left(u^{p\prime}\right) \tag{8}$$
3. solve the following *defect correction* equation in $\mathcal{U}^q$ exactly:
$$\left(\phi_i^q, \mathcal{L}\left(u^q\right) - \mathcal{L}\left(u^{q\prime}\right) + \mathcal{L}\left(u^{p\prime}\right)\right) = 0 \ \ \forall \phi_i^q \in \mathcal{U}^q \tag{9}$$
4. *Prolongate* the correction $u^q - u^{q\prime}$ in $\mathcal{U}^p$ by a suitable operator $\mathcal{T}^{pq}$
$$u^p = u^{p\prime} + \mathcal{T}^{pq}\left(u^q - u^{q\prime}\right) \tag{10}$$
5. Perform a number of additional iterations on level p to smooth the corrected solution.

3.2 Definition of transfer operators

3.2.1 Solution transfer operators

Both the prolongation and restriction operators for the solution are based on the L_2 or Galerkin projection. Let us consider a solution $u^a \in \mathcal{U}^a$ and its projection $\mathcal{T}^{ba}u^a = u^b \in \mathcal{U}^b$

$$\begin{aligned} u^a &= \sum_i \mathbf{u}_i^a \phi_i^a \ , \ \phi_i^a \in \mathcal{U}^a \\ \mathcal{T}^{ba} u^a = u^b &= \sum_j \mathbf{u}_j^b \phi_j^b \ , \ \phi_j^b \in \mathcal{U}^a \end{aligned} \tag{11}$$

Orthogonalising the difference $u^a - u^b$ to the space $\mathcal{U}^b$ defines the following set of equations for the expansion coefficients $\mathbf{u}_i^b$

$$\sum_i \left(\phi_k^b, \phi_j^b\right) \mathbf{u}_j^b = \sum_j \left(\phi_k^b, \phi_i^a\right) \mathbf{u}_i^a, \ \forall \phi_k^b \in \mathcal{U}^b \tag{12}$$

The solution transfer operator $\mathcal{T}^{ba}$ has the discrete or matrix equivalent $\mathbf{T}^{ba}$ defining the transfer between the expansion vectors $\mathbf{u}^a = [\mathbf{u}_1^a ... \mathbf{u}_n^a]^T$ and $\mathbf{u}^b = [\mathbf{u}_1^b ... \mathbf{u}_m^b]^T$

$$\mathbf{u}^b = \mathbf{T}^{ba} \cdot \mathbf{u}^a = \left(\mathbf{M}^{bb}\right)^{-1} \cdot \mathbf{M}^{ba} \cdot \mathbf{u}^a \tag{13}$$

where

$$\mathbf{M}_{ij}^{ab} = \left(\phi_i^a, \phi_j^b\right) \tag{14}$$

3.2.2 Residual transfer operator

The "restriction" of the residual vector follows directly from the weighted defect correction equation 9. Conventionally one goes the other way around: first a restriction operator is defined for the residual vector, and then - in the best of cases - the coarse grid operator is found by applying the discrete transfer operators to the fine grid operator (*Galerkin coarse grid approximation* or *GCGA*) or - more frequently - one uses the same discretisation technique on the coarse representation (*Discrete coarse grid approximation* or *DCGA*). The forcing term contains the "restricted" residual

$$\left(\phi_i^q, \mathcal{L}\left(u^p\right)\right) \tag{15}$$

To compute this term explicitly we would need to redefine routines for weighting all terms of the residual defined in $\mathcal{U}^p$ with shape functions in $\mathcal{U}^q$. Fidkowski[5] developed this residual restriction operator for nested spaces by using the correspondance between the shape functions. This approach may be generalised to non-nested spaces in the following way. First we expand the residual function $\mathcal{L}\left(u^p\right)$ in $\mathcal{U}^p$ using L_2 projection:

$$\begin{gathered} \mathcal{L}\left(u^p\right) \approx \sum_i \mathbf{l}_i^p \phi_i^p \\ \sum_i \mathbf{l}_i^p \left(\phi_i^p, \phi_j^p\right) \approx \left(\phi_j^p, \mathcal{L}\left(u^p\right)\right) = \mathbf{r}_j^p \end{gathered} \tag{16}$$

This projection requires the mass matrix and the Galerkin weighted residual defined on space $\mathcal{U}^p$, both of which are already available. The "restricted residual" is then computed as:

$$\left(\phi_i^q, \mathcal{L}\left(u^p\right)\right) \approx \left(\phi_i^q, \sum r_i^p \phi_i^p\right) \tag{17}$$

We find the following matrix operator connecting the vector containing Galerkin weighted residuals for space $\mathcal{U}^p$ to the restricted residuals in space $\mathcal{U}^q$

$$\tilde{\mathbf{T}}^{qp} = \mathbf{M}^{qp} \cdot \left(\mathbf{M}^{pp}\right)^{-1} \tag{18}$$

Notice that in the case of an orthonormal set of shape functions, the restriction operators for solution and residual coincide.

One sees that all transfer operators necessitate the multiplication of vectors with the inverse of the mass matrices and hybrid mass matrices. The inversion of the mass matrix is in practice only feasible for discontinuous interpolation spaces such as used by DGFEM, since in this case the mass matrix is block-diagonal. Obviously this continues to hold for h-Multigrid on non-conforming meshes.

3.3 Convergence analysis for linear problems and nested interpolations

Suppose we have a linear operator, then we may rewrite the discretised equations as follows:

$$\mathbf{L}^p \cdot \mathbf{u}^p = \mathbf{f}^p \tag{19}$$

where

$$\begin{aligned} \mathbf{L}^p_{ij} &= \left(\phi^p_i, \mathcal{L}\left(\phi^p_j\right)\right) \\ \mathbf{f}^p_i &= \left(\phi^p_i, f\right) \end{aligned} \tag{20}$$

Suppose furthermore that the interpolation space $\mathcal{U}^q$ is nested into $\mathcal{U}^p$, ie. every shape function in $\mathcal{U}^q$ is exactly represented in $\mathcal{U}^p$:

$$\phi^q_i = \alpha^{qp}_{ij} \cdot \phi^p_j \ , \ \forall \phi^q_i \in \mathcal{U}^q \tag{21}$$

This is for instance the case for any hierarchy of Lagrangian interpolation bases defined on the same tesselation, in the case of p-Multigrid, or for equal order interpolations on nested meshes which are found by regular cell divisions in the case of h-Multigrid. Since the mapping is exact, we may find the corresponding projection matrices α^{qp} by any consistent mapping method (eg. L_2 projection):

$$\alpha^{qp} = \left(\mathbf{M}^{qp} \cdot \left(\mathbf{M}^{pp}\right)^{-1}\right) = \tilde{\mathbf{T}}^{qp} = \left(\mathbf{T}^{pq}\right)^T \tag{22}$$

3.3.1 Approximation of the operator on the coarser level

We can now show that the above choice of restriction and prolongation operators are "compatible" to the coarse grid operator. We find:

$$\begin{aligned} \mathbf{L}^q_{ij} &= \left(\phi^q_i, \mathcal{L}\left(\phi^q_j\right)\right) \\ &= \alpha^{qp}_{ik} \cdot \left(\phi^p_k, \mathcal{L}\left(\phi^p_l\right)\right) \cdot \alpha^{qp}_{jl} \\ \mathbf{L}^q &= \alpha^{qp} \mathbf{L}^p \left(\alpha^{qp}\right)^T \\ &= \tilde{\mathbf{T}}^{qp} \cdot \mathbf{L}^p \cdot \mathbf{T}^{pq} \end{aligned} \tag{23}$$

This means that in this case the application of the same discretisation on the coarser level (*DCGA*) results in a discrete operator that coincides with the Galerkin Coarse Grid Approximation (*GCGA*).

3.3.2 Two-level cycle convergence analysis

A consequence of Eq. (23) is that we may reorganise the defect correction equation as

$$\tilde{\mathbf{T}}^{qp}\left(\mathbf{L}^p \cdot \left(\mathbf{u}^{p\prime} + \mathbf{T}^{pq} \cdot \left(\mathbf{u}^q - \mathbf{u}^{q\prime}\right)\right) - \mathbf{f}^p\right) = 0 \tag{24}$$

Eq. (24) explicitly states that the residual vector in the space $\mathcal{U}^p$ after prolongation of the coarse level correction is in the nullspace of the residual restriction matrix. This means that another two-level iteration starting from the corrected solution will no longer lead to any correction, avoiding limit-cycles due to multigrid. Notice that the choice of the solution restriction operator is not of importance for the linear case; obviously it remains important for the non-linear case.

Now we can adapt the two-grid analysis from Wesseling[12] . The error vector $\mathbf{e}^p$ is defined as the difference between the current solution vector $\mathbf{u}^p$ and the final solution $\mathbf{u}^{p*}$, and satisfies the following equation:

$$\mathbf{L}^p\mathbf{e}^p = \mathbf{L}^p\mathbf{u}^p - \mathbf{f}^p = \mathbf{r}^p \tag{25}$$

The defect correction equation now becomes:

$$\tilde{\mathbf{T}}^{qp}\left(\mathbf{L}^p \cdot \left(\mathbf{e}^{p\prime} + \mathbf{T}^{pq} \cdot \left(\mathbf{u}^q - \mathbf{u}^{q\prime}\right)\right)\right) = 0 \tag{26}$$

The error $\mathbf{e}^p$ after one two-level iteration now becomes

$$\mathbf{e}^p = \left(\mathbf{I}^p - \mathbf{T}^{pq} \cdot \left(\mathbf{L}^q\right)^{-1} \cdot \tilde{\mathbf{T}}^{qp} \cdot \mathbf{L}^p\right) \cdot \mathbf{e}^{p\prime} \tag{27}$$

We may now decompose the error $\mathbf{e}^{p\prime}$ into a *smooth* component ${\mathbf{e}^{p\prime}}_S$ belonging to the range of $\mathbf{T}^{pq}$, and a *rough* part ${\mathbf{e}^{p\prime}}_S$ belonging to the kernel of $\left(\mathbf{T}^{pq}\right)^T = \tilde{\mathbf{T}}^{qp}$ and orthogonal to the range of $\mathbf{T}^{pq}$.

$$\begin{aligned} {\mathbf{e}^{p\prime}}_S &= \left(\mathbf{T}^{pq} \cdot \left(\tilde{\mathbf{T}}^{qp}\mathbf{T}^{pq}\right)^{-1} \cdot \tilde{\mathbf{T}}^{qp}\right) \cdot \mathbf{e}^{p\prime} \\ {\mathbf{e}^{p\prime}}_R &= \left(\mathbf{I}^p - \mathbf{T}^{pq} \cdot \left(\tilde{\mathbf{T}}^{qp}\mathbf{T}^{pq}\right)^{-1} \cdot \tilde{\mathbf{T}}^{qp}\right) \cdot \mathbf{e}^{p\prime} \end{aligned} \tag{28}$$

we find that the error after the coarse grid correction only depends on the rough or high order part of the initial error

$$\mathbf{e}^p = \left(\mathbf{I}^p - \mathbf{T}^{pq} \cdot \left(\mathbf{L}^q\right)^{-1} \cdot \tilde{\mathbf{T}}^{qp} \cdot \mathbf{L}^p\right) \cdot {\mathbf{e}^{p\prime}}_R \tag{29}$$

This does not mean however that the resulting error is outside of the range of $\mathbf{T}^{pq}$, or - otherwise stated - is entirely made up of “rough” components, this in contrast to the residual.

3.3.3 Correspondence of eigenspaces

In the case of finite difference type discretisations, the eigenspace of the discretised operator consists of Fourier modes. When applying h-Multigrid to these methods a clear distinction may be found between fine and coarse grid eigenmodes, since 'coarse' eigenmodes are 'fine' eigenmodes as well.

For higher-order DGFEM, the eigenspace of the discrete operator on the coarse and the fine representation is different. Therefore - in our opinion - a Fourier analysis[7] to determine convergence rate is not applicable. If an exploitable correspondance between the eigenvector spaces of the fine and the coarse level operators could be found, we could split up the eigenspace into rough and smooth modes, and quantify the ideal efficiency of the method. We have not yet been able to find such a decomposition for p-Multigrid DGFEM, but are currently working on this issue.

3.4 Multigrid strategies

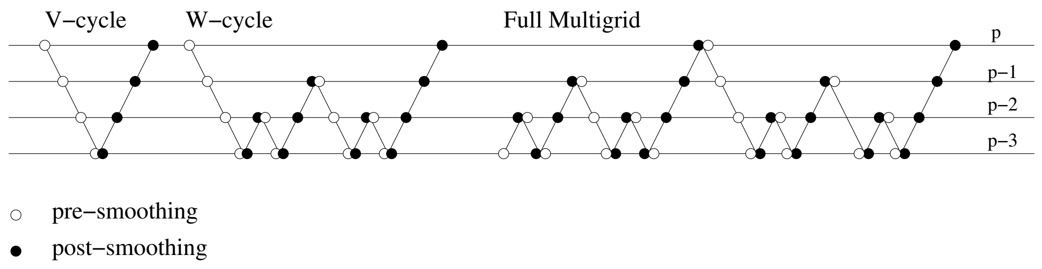


Figure 2: Multigrid strategies

The two-grid algorithm requires an accurate solve of the defect correction equation. This may still be untractable, and hence a number of strategies exploiting the two-grid cycle are possible.[12] Two types of strategies have been studied (see Fig. 2):

- *V- and W-cycles* consist in replacing the exact solution of the defect correction equation on the coarse level by a recursive application of the two-grid cycle. Depending on the number of two-cycle solves per defect correction we distinguish v-cycles and w-cycles. We define our multigrid cycling strategy by the following parameters
  - number of coarser levels
  - number of pre- and post-smoothing steps (or accuracy) for each level
  - number of subcycles
  - smoother type for each level

  Typically we visit all available levels.

- *Full multigrid (FMG)* or *Nested Iteration* starts with iterations on the coarsest representation. After that we successively refine the representation. Using the hitherto available coarser levels one performs a number of v- or w-cycles. Depending on the strategy a prespecified number of cycles is done, or until a desired level of convergence has been reached. After that the solution is prolongated to the next finer level. When finally reaching the target order, one continues v- or w-cycling until full convergence.

4 HYBRID ITERATION STRATEGY

4.1 Elementary iterative methods

On the one hand, we use a local time-step variant of the classical 4-step Runge-Kutta method, which is fairly often used owing to the classical application of DGFEM to time-accurate integration. This method has a minimal impact on memory footprint, but is extremely slow to converge.

On the other hand we have implemented the 'matrix-free' Newton-Krylov-ILU scheme, which is fairly often used in finite volume methods[3,8] and has only recently been applied to DGFEM.[1,10] In this method the discretised non-linear set of equations is solved using Newton iterations. The resulting linear system at each Newton step is iteratively solved with a Krylov subspace iteration method. In this case we use the classical combination of GMRES with ILU(0) as preconditioner; the matrix-vector product needed in GMRES is provided by finite difference. Furthermore we add a pseudo-time derivative at each iteration step n:

$$\left(\psi^p, \frac{(u^{p,n} - u^{p,n-1})}{\Delta\tau^n} + \nabla \cdot \vec{\mathcal{F}}\left(u^{p,n}\right)\right) = 0 \ , \ \forall \psi^p \in \mathcal{U}^p \tag{30}$$

The pseudo-time step $\Delta\tau^n$ is chosen locally to conform to a given CFL number, and thus provide the “locally optimal” diagonal dominance for the linear solution as well as a sufficient underrelaxation to avoid non-linear excursions. As the solution converges, the need for stabilisation diminishes. Therefore the CFL number is updated following each Newton update, inversely proportional to a power α of the current reduction of the L_2 norm of the residual ϵ:

$$CFL_n = max\left(CFL_0 \cdot \left(\frac{\|\epsilon_0\|_2}{\|\epsilon_n\|_2}\right)^{\alpha}, CFL_{max}\right) \tag{31}$$

The main advantage of the Newton method is its computational efficiency. Its main drawback for high-order DGFEM is the impressive memory footprint. The elementary blocks composing the preconditioning matrix increase quadratically in size with the number of unknowns per element/face. If p is the polynomial order, then the number of unknowns per element grows like p^2 in 2 and p^3 in 3 dimensions. Then, the number of elements in each block of the ILU grows like p^4 resp. p^6. Hence a matrix preconditioner becomes quite impractical even for relatively low orders.

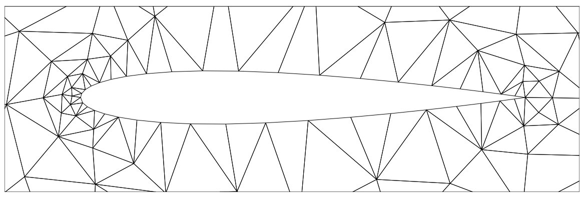
4.2 Hybrid p-Multigrid

Multigrid convergence hinges upon the convergence of the lowest frequency(order) errors, which are no longer converted to high frequencies by coarsening. Hence, to obtain ideal multigrid convergence, we need in theory to solve the coarsest level exactly at each cycle, or in multigrid terminology, we need a smoother that is also effective on long-wave, low frequency errors. On the finer levels the smoother may only be efficient on the rough parts of the residual.

On the one hand, we know that the Newton-Krylov implicit solver is highly effective in removing errors, more or less independent of frequency content, but the storage needed for the ILU preconditioner only remains reasonable on the coarsest levels of interpolation. On the other hand the Runge-Kutta scheme does a reasonable job at removing higher order or rough parts of the residual but has only a very small memory footprint.

Combining the above reflections we may conclude that the combination of a Newton iterator on the coarsest level with Runge-Kutta or equivalently simple smoothers on the finer levels may be very competitive. If we furthermore consider that extremely coarse meshes are used to represent the solution, we may conclude that a p-Multigrid method using the implicit scheme on the coarsest level will be very memory-efficient with respect to conventional CFD codes. Another advantage is that the classical slope-limiter for DGFEM[4] is not easily integrated into an implicit iteration method.

5 COMPARISON OF ITERATIVE STRATEGIES


(a) Coarse resolution

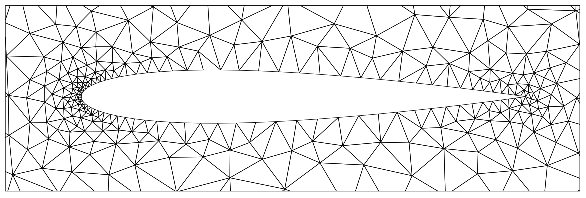
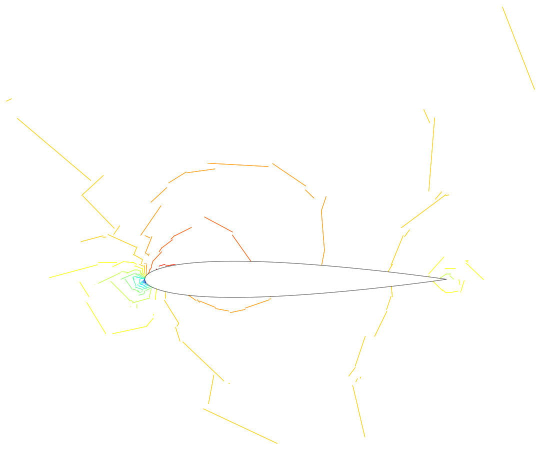

(b) Fine resolution

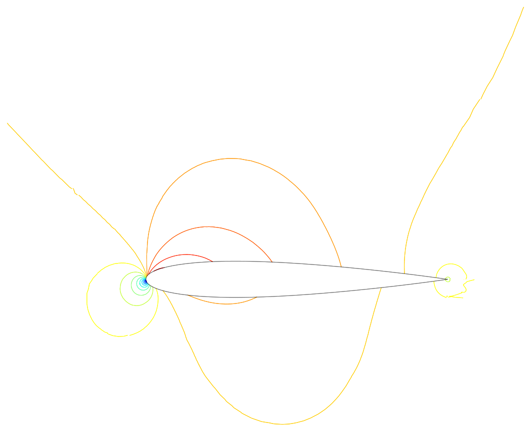
Figure 3: Mesh details for the flow around the NACA0012 airfoil

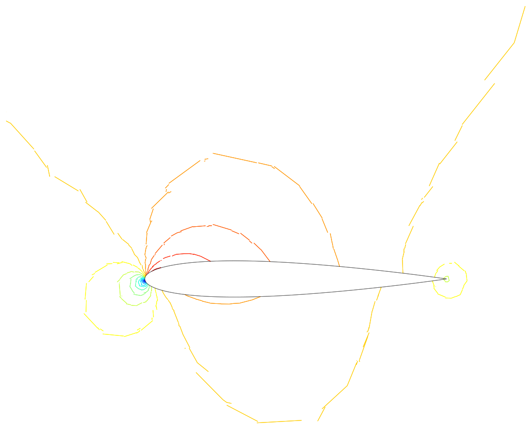
To compare the iteration strategies presented in this paper we compute the inviscid flow of air around the NACA0012 airfoil. The angle of attack is 2^o and the Mach number is 0.3. The chord has been rescaled to avoid the truncation of the airfoil at the trailing edge. The free-stream boundary is located at 30 chords from the leading edge. The mesh size on the airfoil varies quadratically, from 0.01 at the leading edge, over 0.08 at midchord to 0.02 at the trailing edge. At the free-stream boundary the grid size is 20. The coarse mesh is obtained by doubling those sizes. The details of the meshes near the airfoil are shown in Fig.3. The coarse mesh contains in total 157 nodes, the fine mesh 622 nodes.

At all explicit levels use 10 Runge-Kutta pseudo-time iterations for both pre- and post-smoothing sweeps, with $CFL = 2$. The implicit Newton-Krylov solver starts out with $CFL_0 = 5$; the exponent $\alpha = 0.5$. 30 Krylov vectors are used, without restart. The initial CFL number CFL_0 for the Newton-Krylov scheme is updated following each multigrid cycle, by using the same rationale as in (31), but now using the ratios of fine grid residual norms. If the Runge-Kutta and Newton-Krylov method are used outside of the multigrid framework, the same parameters are applied.

The computed Mach number isolines for interpolation orders 1 and 4 on the coarse and on the fine mesh are compared in Fig.4.


(a) p=1, coarse


(b) p=4, coarse


(c) p=1, fine

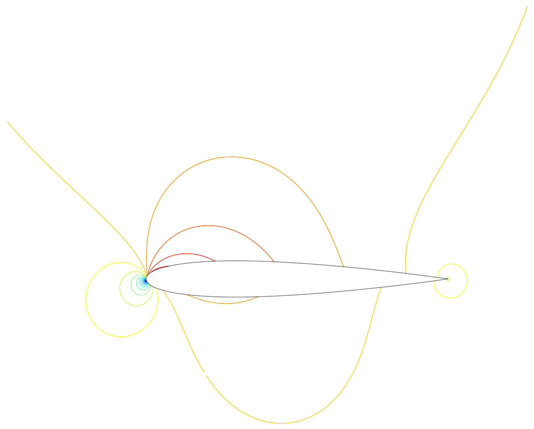
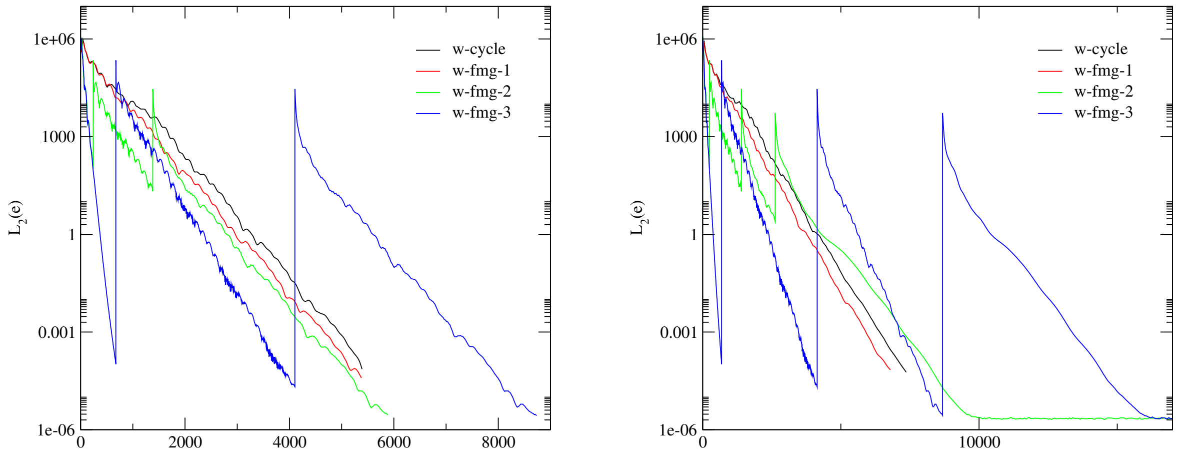

(d) p=4, fine

Figure 4: Mach number distribution

In Fig. 5 we compare performance of three full multigrid strategies in terms of CPU time. All strategies use fully explicit w-cycles and are compared to the standard w-cycle. Residuals are always computed using the solution projected onto the highest order representation. The first strategy, labeled "w-fmg1" only uses one cycle per level during the nested iteration. The second one ("w-fmg2") converges 4 orders of magnitude for each refinement. The last one ("w-fmg3") converges each level to machine accuracy. We see that w-fmg1 and w-fmg2 have more or less the same performance as the w-cycle. W-fmg3 performs much worse. From a given resolution onward the low order solution we obtain generates the same higher-order error. All the time spent in improving the low order solution any further is then lost. At least for this case, we have nothing to gain from FMG. The only advantage we may reasonably expect from FMG is an increased stability.



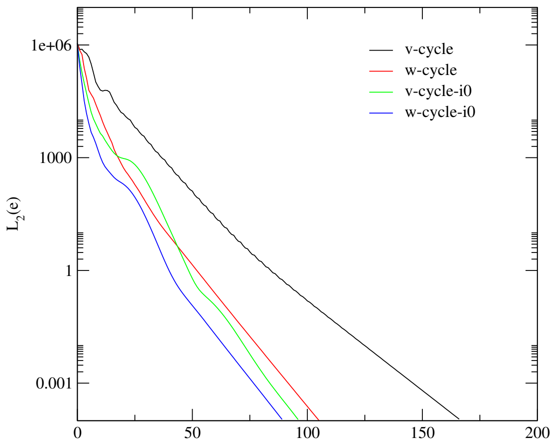
(a) order=2, fine mesh

(b) order=3, fine mesh

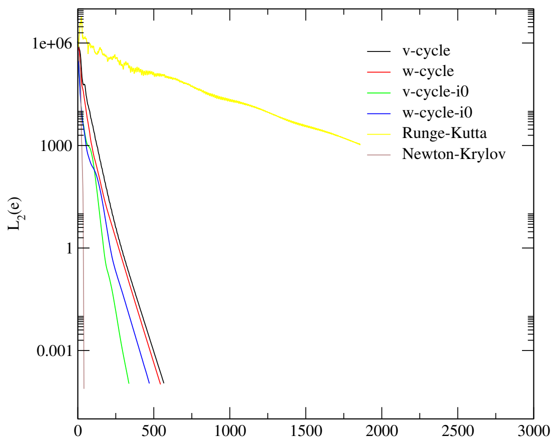
Figure 5: Comparison of FMG strategies, fine mesh

In Figs. 6 and 7 we compare the convergence rates of the different iteration strategies in terms of number of multigrid cycles (if applicable) and CPU time for orders 2, 3 and 4 on the coarse and the fine mesh respectively. The strategies included in this study are:

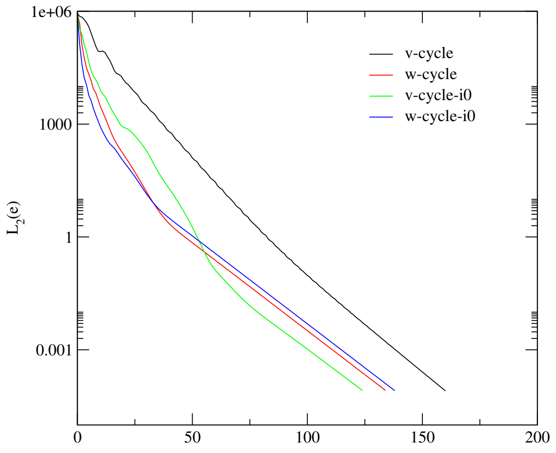
- fully explicit v-cycle (labeled "v-cycle")
- fully explicit w-cycle (labeled "w-cycle")
- v-cycle with implicit coarsest level (labeled "v-cycle-i0")
- w-cycle with implicit coarsest level (labeled "w-cycle-i0")
- Runge-Kutta pseudo-timestepping (labeled "Runge-Kutta")
- Newton-Krylov-ILU (labeled "Newton-Krylov")



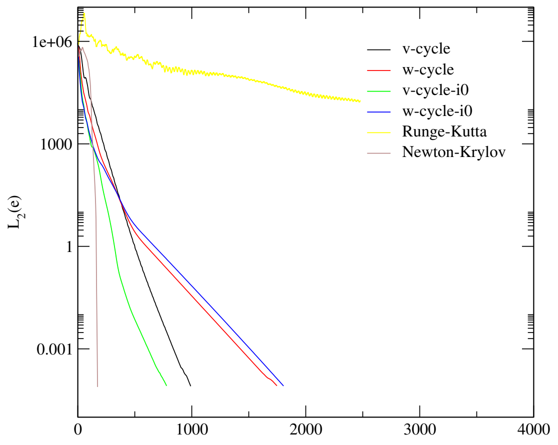
(a) Number of cycles, order=2



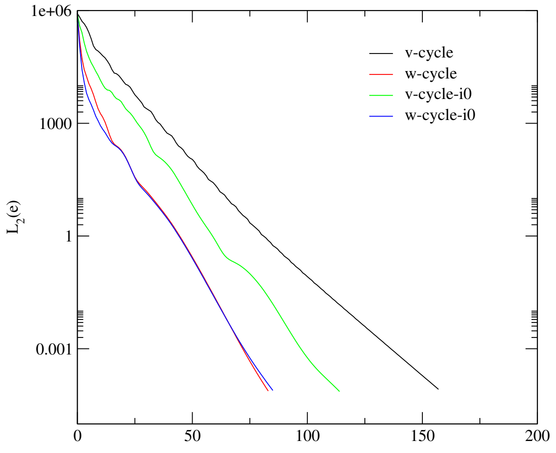
(b) CPU time, order=2



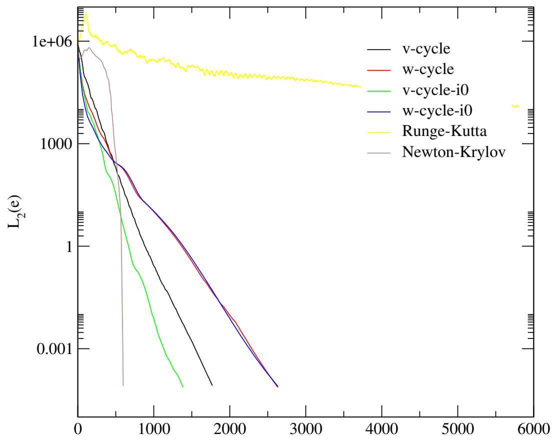
(c) Number of cycles, order=3



(d) CPU time, order=3

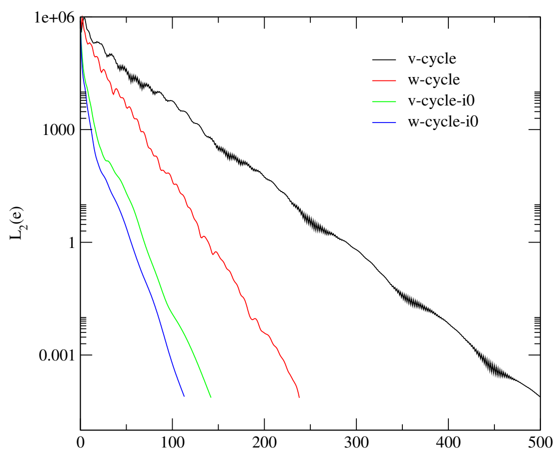


(e) Number of cycles, order=4

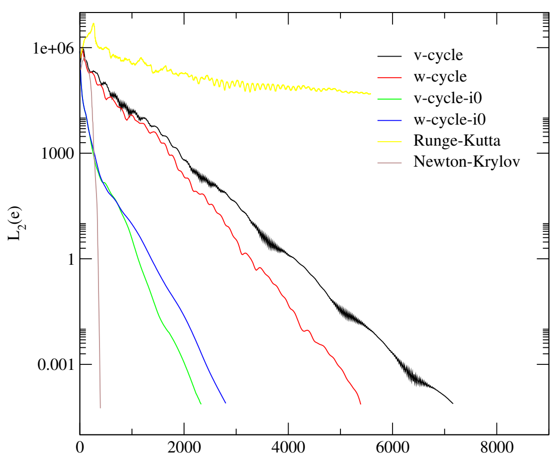


(f) CPU time, order=4

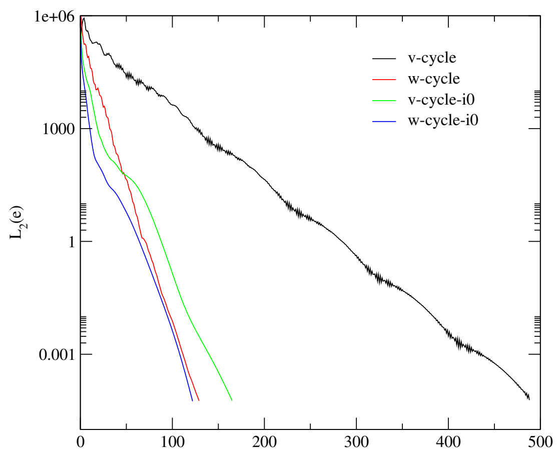
Figure 6: Comparison of iteration strategies, coarse mesh



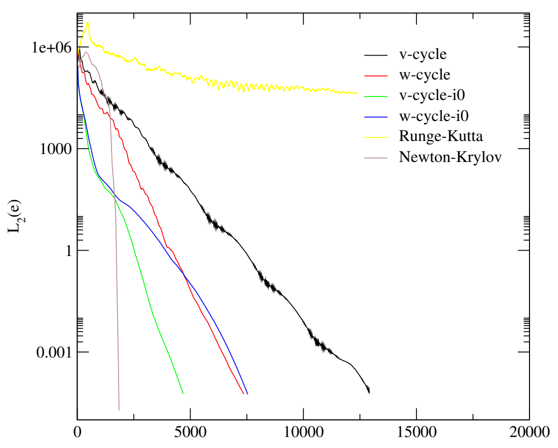
(a) Number of cycles, order=2



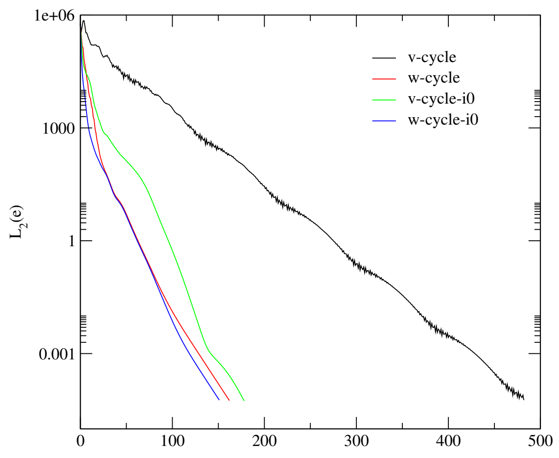
(b) CPU time, order=2



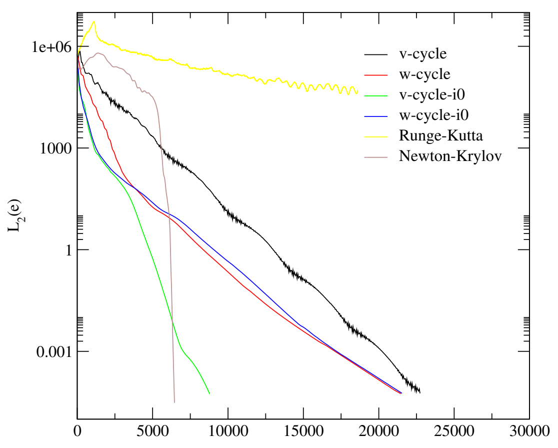
(c) Number of cycles, order=3



(d) CPU time, order=3



(e) Number of cycles, order=4



(f) CPU time, order=4

Figure 7: Comparison of iteration strategies, fine mesh

We see that the v-cycles have the same rate of convergence per cycle irrespective of the finest-level interpolation order, i.e. we attain textbook multigrid efficiency. We see this behaviour only for v-cycles since the number of passes on each level of this strategy is independent of the finest-level interpolation order. The number of cycles however is very dependent on the grid size. This is to be expected, since the maximum stable timestep is proportional to the grid size. If we had used a classical multigrid method, using coarser grids instead of decreasing order, we would have convergence rates per cycle independent of mesh size. A combination of p- and classical multigrid could provide both order and grid size independent convergence rates.

We note that in any configuration (fully or partly explicit) w-cycles have a higher asymptotic convergence rate per cycle, almost in a constant ratio to the number of coarser level sweeps. Since the extra work is mostly confined to the coarser levels we can sometimes see a better performance in terms of CPU time.

The application of the Newton-Krylov strategy as the smoother on the coarsest level tends to speed up v-cycles considerably, both in number of cycles and CPU time. For w-cycles the effect is most marked in the early stages, where the convergence rate seems to be dominated by low frequency errors. After an initial transient however we find asymptotic convergence rates that are similar to the ones for the fully explicit cycles. A possible explanation is that w-cycles, as they concentrate most of the workload on the coarser levels, smooth out the low-frequency error more effectively. Consequently this strategy, especially in the partly implicit case, would be dominated by the convergence of higher frequencies only in the last stage of convergence. V-cycles would tend to keep a balanced "broad-band" error, and maintain efficiency on all levels up to the end.

Finally we also note that in all cases the implicit Newton-Krylov scheme converges first. However as order and number of DOFs increases, the multigrid scheme becomes a viable competitor, even when we use a poor smoother such as Runge-Kutta, and this for a fraction of the memory.

6 CONCLUSIONS

First the framework for the application of multigrid to DGFEM has been presented and analysed. After this an efficient hybrid p-Multigrid iteration strategy for DGFEM has been presented and compared to standard iteration strategies. Even using very simple smoothers, the hybrid p-Multigrid method proves to be a performant option in terms of CPU time whilst drastically reducing memory footprint and code complexity.

7 ACKNOWLEDGMENTS

CENAERO is funded by the Walloon Region and the structural funds ERDF and ESF (EP1A122030000102).

Grids and visualisations were generated with Gmsh (http://www.geuz.org/gmsh), one-dimensional graphs with Grace (http://plasma-gate.weizmann.ac.il/Grace).

REFERENCES

[1] Bassi, F. and Rebay, S. GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations *Discontinuous Galerkin methods: theory, computation and applications*, Lecture Notes in Computational Science and Engineering, vol 11, Springer-Verlag (2000).

[2] Bastian P. and Reichenberger, V. *Multigrid for higher order discontinuous Galerkin finite elements applied to groundwater flow*, Technical report SFB 359, Interdisziplinäres zentrum für Wissenschaftliches Rechnen, Universität Heidelberg (2000).

[3] Brown, P.N. and Saad, Y. Hybrid Krylov methods for nonlinear systems of equations. *SIAM Journal on Scientific and Statistical Computing*, **11**(3):450–481 (1990).

[4] Cockburn, B. and Shu, C.-W. *The Runge-Kutta Discontinuous Galerkin Method for conservation laws V : multidimensional systems*, Journal of Computational Physics, **141**:199–224 (1998).

[5] Fidkowski, K.J. and Darmofal, D.L. *Development of a higher-order solver for aerodynamic computations*, 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, 2004 (paper 2004-0436).

[6] Hartmann, R., *Adaptive discontinuous Galerkin methods with shock-capturing for the compressible Navier-Stokes equations*, to be published in International Journal for Numerical Methods in Fluids.

[7] Helenbrook, B.T., Mavriplis, D. J. and Atkins, H. A *Analysis of p-multigrid for continuous and discontinuous finite element discretizations*, 16th AIAA Computational Fluid Dynamics Conference, Orlando, Florida, 2003 (paper 2003-3989).

[8] Keyes, D.E. and Venkatakrishnan, V. *Newton-Krylov-Schwarz methods: interfacing sparse linear solvers with nonlinear applications*, Zeitschrift für Angewandte Mathematik und Mechanik, **76**:147–150 (1996).

[9] Luo, H., Baum, J.D. and Löhner, R. *A p-Multigrid discontinuous Galerkin Method for the compressible Euler equations on unstructured grids*, Finite Elements for Flow Problems, Swansea, 2005.

[10] Rasetarinera, P. and Hussaini, M.Y. *An efficient implicit discontinuous spectral Galerkin method.* Journal of Computational Physics **172**:718–738 (2001).

[11] van der Vegt, J.J.W. and van der Ven, H., *Space time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows: I. General formulation*, Journal of Computational Physics **182**:546-585 (2002).

[12] Wesseling G. *An introduction to multigrid methods*, John Wiley and Sons (1991).