Chapter 21
Agglomeration Multigrid for the Vertex-Centered Dual Discontinuous Galerkin Method

Sven-Erik Ekström and Martin Berggren

Abstract. Agglomoration multigrid is used in many finite-volume codes for aerodynamic computations in order to reduce solution times. We show that an existing agglomeration multigrid solver developed for equations discretized with a vertex-centered, edge-based finite-volume scheme can be extended to accelerate convergence also for a vertex-centered discontinuous Galerkin method. Preliminary results for a subsonic as well as a transonic test case for the Euler equations in two space dimensions show a significant convergence acceleration for the discontinuous Galerkin equations using the agglomoration multigrid strategy.

1 Introduction

The particular discontinuous Galerkin (DG) method studied and implemented by Uppsala University within the ADIGMA project is vertex centered, in contrast to the standard cell-centered DG method. The method was introduced by Berggren *et al.* [1, 2] for model problems and for the Euler equation in our other contribution to this volume [4]. The method is designed to constitute a generalization to higher order of the edge-based, vertex-centered finite volume (FV) discretization that is particularly popular in many of the codes in engineering practice. Our implementation is done within a software system of this type, Edge [5, 6].

The use of multigrid is a common and often successful strategy for convergence acceleration in FV solvers. We here describe how an existing agglomerated

Sven-Erik Ekström
Department of Information Technology, Uppsala University, Box 337, SE-751 05 Uppsala, Sweden
e-mail: sven-erik.ekstrom@it.uu.se

Martin Berggren
Department of Computing Science, Umeå University, SE-901 87 Umeå, Sweden
e-mail: martin.berggren@cs.umu.se

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 301–308, 2010.
springerlink.com

multigrid facility in Edge is adapted to support multigrid also for the DG discretization. To explain the multigrid approach, we adopt a notation similar to the one used in the Edge documentation [6].

We want to solve the steady state problem

$$\nabla \cdot \mathscr{F}(U) = 0 \quad \text{in } \Omega, \tag{1}$$

where $\mathscr{F}$ is the standard flux function for the Euler equations [3, 4], and $U = [\rho, \rho\mathbf{u}, \rho E]^T$ of dimension $d+2$ (d is the space dimension) is the vector of conservative variables. Let each components of U_h, the numerical solution to equation (1), belong to the space V_h specified in [4]. The DG method is obtained by multiplying equation (1) by a test function vector $V_h \in \mathscr{V}_h^{d+2}$, integrating over each control volume, integrating by parts, and introducing the numerical flux $\mathscr{F}^*$ on the boundaries. This procedure yields that $U_h \in \mathscr{V}_h^{d+2}$ solves the variational problem

$$\int_{\partial K_m} V_h \cdot \mathscr{F}^*(U_L, U_R, \hat{\mathbf{n}})\,\mathrm{d}s - \int_{K_m} \nabla V_h \cdot \mathscr{F}(U_h)\,\mathrm{d}V = 0, \ \forall K_m \subset \Omega, \quad \forall V_h \in \mathscr{V}_h^{d+2}, \tag{2}$$

where subscripts L and R denote local ("left") and remote ("right") values on the boundary ∂K_m of control volume K_m, and $\hat{\mathbf{n}}$ is the outward unit normal.

Variational expression (2) defines a nonlinear equation $\mathscr{N}(U_h) = 0$. To solve this equation, a common approach in the the CFD community is to time march equation

$$\frac{\partial U_h}{\partial t} + \mathscr{N}(U_h) = 0 \tag{3}$$

to steady state using Runge–Kutta time stepping. For efficiency, this procedure needs to be accelerated by for example local time-stepping and multigrid, the latter which is the subject of this chapter. We start in Section 2 with a brief description of the existing agglomeration multigrid method as implemented in Edge. In Section 3, we explain how the method has been extended to encompass also the vertex-centered DG discretization. Sections 4 presents preliminary results of our DG agglomoration multigrid approach, and we end with some concluding remarks in Section 5.

2 Existing Finite Volume Agglomeration Multigrid in Edge

For use in the agglomoration multigrid process, the preprocessor of Edge generates a sequence of L coarser and coarser dual meshes. First, a dual mesh (top right in Figure 1) is generated from the primal mesh (top left in Figure 1). The data structures needed to represent the dual mesh is a list of edges connecting adjacent vertices in the primal mesh and a list, associated with these edges, of normals to the dual control volumes. (Additional boundary information is also required [4], but for simplicity not considered here).

The meshes are indexed coarse to fine by $1, \ldots, L$, where mesh L refers to the dual mesh generated from the supplied primal mesh. To generate a coarser mesh, a set of adjacent dual cells are agglomerated into bigger cells, as shown in the bottom left of Figure 1; this mesh would be number $L-1$. The number of degrees of freedom is equal to the number of cells, hence less for the coarser mesh, and the location of each new node is computed by a weighted average of the nodes in the dual cells (of the finer mesh) that constitute the agglomerated cell. A new list of edges connecting nodes in the agglomerated mesh is generated, and new normals are constructed by vectorially adding normals from the finer mesh. In the bottom right of Figure 1, yet another agglomeration is shown, with even fewer elements, using the same procedure to generate a new list of edges and normals; this mesh would be numbered $L-2$.

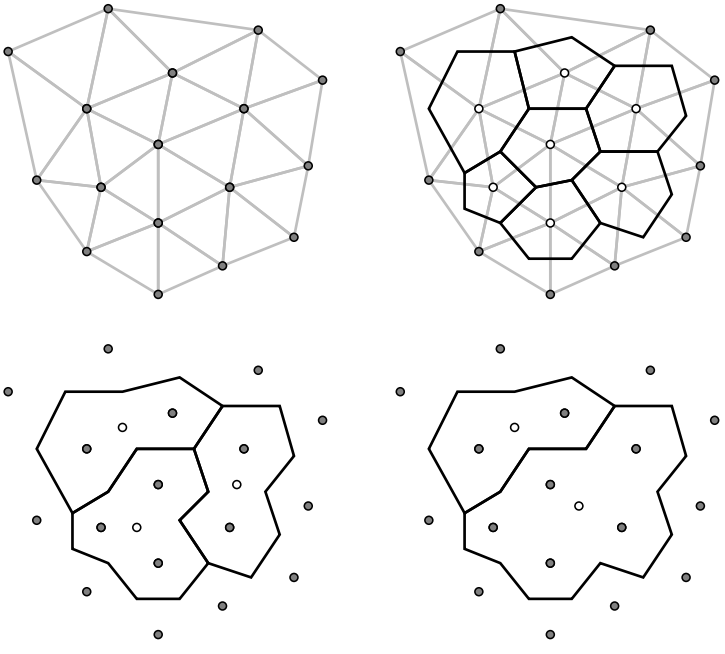

Fig. 1 The preprocessor stages to generate computational meshes. Top left: The primal mesh. Vertices are marked with gray circles. Top right: The dual mesh. The degrees of freedom for the unknowns are associated with the nodes (located at the primal mesh vertices) marked with white circles. The data structures involve lists, associated with the edges of the primal mesh, of (pair of) node numbers and the normals of the dual cells. This is level L in the agglomerated multigrid cascade of L meshes. Bottom left: Agglomeration one step, combining adjacent dual cells to bigger cells. The (white) node in each agglomerated cell is weighted together from the (gray) nodes in the finer cells constituting the agglomerated cell. A single normal for each edge in the agglomerated mesh is constructed by vectorially adding normals from the associated part of the cell boundary. This level would be $L-1$. Bottom right: Agglomeration one more level with even bigger cells, at level $L-2$.

The discrete Euler or Navier–Stokes equations on the finest mesh can be written

$$\frac{\partial U^L}{\partial t} + \mathcal{N}_L(U^L) = 0, \tag{4}$$

where $\mathcal{N}_L(U^L)$ is the spatial discretization of the equations on the finest mesh, here the FV discretization (or lowest order DG, $p = 0$). At each coarser mesh level $l < L$, we have

$$\frac{\partial U^l}{\partial t} + \mathcal{N}_l(U^l) = F_l, \tag{5}$$

with initial value $U^l = I^l_{l+1}U^{l+1}$, where I^l_{l+1} is the restriction operator for the unknowns, and F_l is a forcing function, defined recursively as

$$F_l = \mathcal{N}_l(I^l_{l+1}U^{l+1}) + \hat{I}^l_{l+1}\left[F_{l+1} - \mathcal{N}_{l+1}(U^{l+1})\right], \tag{6}$$

with $F_L = 0$. Moreover, I^l_{l+1} and $\hat{I}^l_{l+1}$ are restriction operators of respectively the unknowns and the residuals from finer mesh $l+1$ to coarser mesh l. These are defined cell by cell by summing contributions from the subcells on mesh level $l+1$ that constitute the cell on mesh level l. The restriction of $\hat{I}^l_{l+1}$ to cell K_m at mesh level l is given by

$$\hat{I}^l_{l+1}|_m R^{l+1} = \sum_n R^{l+1}_n, \tag{7}$$

where R^{l+1} is the vector of residuals with components R^{l+1}_n associated with cell K_m. The restriction of I^l_{l+1} to cell K_m is similarly defined, but with a weighting involving the cell volumes:

$$I^l_{l+1}|_m U^{l+1} = \frac{\sum_n \mathcal{V}^{l+1}_n U^{l+1}_n}{\sum_n \mathcal{V}^{l+1}_n} \tag{8}$$

in which $\mathcal{V}^{l+1}_n$ is the cell volume of a subcell to K_m.

Smoothing is accomplished at each level by integrating (4) and (5) with a few Runge–Kutta steps, typically with local time stepping. When the solution has been smoothed on coarsest mesh, $l = 1$, the following prolongation scheme successively provides updated solutions $\bar{U}^l$, for $l > 1$:

$$\bar{U}^l = U^l + I^l_{l-1}(U^{l-1} - I^{l-1}_l U^l). \tag{9}$$

Operator I^l_{l-1} is the prologation operator (a simple injection operator), from mesh level $l-1$ to mesh level l. For the standard V, W, and F cycle schemes, smoothing is performed on the updated solution $\bar{U}^l$ before proceeding to even finer meshes.

Different choices of fluxes can be chosen by the user on coarser meshes to reduce the computational complexity for each multigrid sweep.

Full multigrid is also available within Edge. The calculations then start at the coarsest mesh and continue until convergence, that is, until the residual is lower

than a user-supplied threshold. After this, the solution is prolonged to the next finer mesh level. Two-grid cycles are now used until convergence is attained again. Then the solution is prolonged to the third mesh level and a three-grid cycle is used until convergence. This procedure is repeated until all meshes are involved and solution is given on the finest mesh. User-defined variables control the behavior of the solver, for example the maximum number of cycles spent in each stage of the full multigrid.

3 Discontinuous Galerkin Agglomeration Multigrid

In this section, we discuss how we adapted the Edge code to support multigrid also for the DG discretization.

The equation to solve at the finest mesh level L and for order p is

$$\frac{\partial U_p^L}{\partial t} + \mathscr{N}_L^p(U_p^L) = 0, \tag{10}$$

where $\mathscr{N}_L^p$ is the spatial discretization operator defined by equation (2). The multigrid process acts through repeated smoothing, that is, by applying a few Runge–Kutta iteration steps on equation (10), with successively modified initial conditions. These initial conditions are obtained recursively by similar smoothing procedures on lower-order approximations and on cruder meshes.

Assume now that $\bar{U}_p^L$ is the result of applying a few Runge–Kutta iteration steps to equation (10). At next multigrid level, we integrate on the same mesh, but at lowest order $p = 0$:

$$\frac{\partial U_0^L}{\partial t} + \mathscr{N}_L(U_0^L) = F_L^0, \tag{11}$$

where

$$F_L^0 = \mathscr{N}_L^0(J_p^0 \bar{U}_p^L) - \hat{J}_p^0\left[\mathscr{N}_L^p(\bar{U}_p^L)\right], \tag{12}$$

and where J_p^0 and $\hat{J}_p^0$ are the restriction operators for the solution and the residual, respectively, as defined below. Since $p = 0$ is nothing else but an vertex-centered finite-volume scheme, equation (11) can be integrated using the agglomeration multigrid scheme in Edge outlined in section 2. Let $\bar{U}_0^L$ be the result of such an integration. Then we may define

$$\bar{\bar{U}}_p^L = \bar{U}_p^L + J_0^p\left(\bar{U}_0^L - J_p^0 \bar{U}_p^L\right) \tag{13}$$

and use $\bar{\bar{U}}_p^L$ as a new initial condition for equation (10). The prolongation operator J_0^p is specified below. For orders $p > 1$, it may also be advantageous to add a level of p-multigrid instead of projecting directly on $p = 0$ as described above. We have not yet implemented or tested such a general p multigrid strategy.

In order to specify the restriction and prolongation operators, let

$$U_q^L|_m = \underbrace{\begin{pmatrix} \rho_1^m & (\rho\mathbf{u})_1^m & (\rho E)_1^m \\ \rho_2^m & (\rho\mathbf{u})_2^m & (\rho E)_2^m \\ \vdots & \vdots & \vdots \\ \rho_{N_m}^m & (\rho\mathbf{u})_{N_m}^m & (\rho E)_{N_m}^m \end{pmatrix}}_{[N_m \times d+2]}, \tag{14}$$

be the piece, associated with macro element K_m, of the solution vector $U_q^L$of order q at the finest mesh level L. The number of degrees of freedom N_m depends on the order; $N_m = 1$ for $q = 0$, for instance. The restriction operator J_p^0 for the solution is block diagonal on each piece $U_p^L|_m$. The diagonal blocks $J_p^0|_m$ are defined by

$$U_0^L|_m = \underbrace{(\mathbf{M}^{00})^{-1}\mathbf{M}^{0p}}_{J_p^0|_m} U_p^L|_m \tag{15}$$

where $\mathbf{M}^{00}$ and $\mathbf{M}^{0p}$ are the mass matrices

$$\mathbf{M}^{00} = \int_{K_m} \mathrm{d}V, \tag{16}$$

$$\mathbf{M}^{0p} = \underbrace{\left(\int_{K_m} \phi_1 \,\mathrm{d}V, \ldots, \int_{K_m} \phi_{N_m} \,\mathrm{d}V \right)}_{[1 \times N_m]}. \tag{17}$$

The restriction operator for the residuals, $\hat{J}_p^0$, is also block diagonal. The diagonal blocks $\hat{J}_p^0|_m$ are defined by

$$\hat{J}_p^0|_m = \mathbf{M}^{0p} \left(\mathbf{M}^{pp}\right)^{-1}, \tag{18}$$

where

$$\mathbf{M}^{pp} = \underbrace{\begin{pmatrix} \int_{K_m} \phi_1 \phi_1 \,\mathrm{d}V & \ldots & \int_{K_m} \phi_1 \phi_{N_m} \,\mathrm{d}V \\ \vdots & & \vdots \\ \int_{K_m} \phi_{N_m} \phi_1 \,\mathrm{d}V & \ldots & \int_{K_m} \phi_{N_m} \phi_{N_m} \,\mathrm{d}V \end{pmatrix}}_{[N_m \times N_m]}. \tag{19}$$

Finally, each diagonal block of the prolongation operator J_0^p, is given by

$$J_0^p = \left(\mathbf{M}^{pp}\right)^{-1} \mathbf{M}^{p0}, \tag{20}$$

where $\mathbf{M}^{p0} = \left(\mathbf{M}^{0p}\right)^T$.

4 First Results

In Figures 2 and 3, we show results for two Mandatory Test Cases of the ADIGMA project, MTC1 and MTC2. The left displays in Figures 2 and 3 depict the pressure coefficient for linear elements for the respective test cases. (The oscillations in the pressure coefficient at the leading edge for the subsonic case vanish when using higher-order elements, as shown in our other contribution to this volume [4].) The right displays in Figures 2 and 3 show the iteration histories, in terms of the mass conservation residuals, when solving MTC1 and MTC2 without and with multigrid. The multigrid computations are initiated with FV solutions obtained with full multigrid, and two grid levels are used in addition to the projection from $p = 1$ to $p = 0$. A clear reduction of number of iteration is attained in both test cases, although the code has not yet been tuned and optimized in any way.

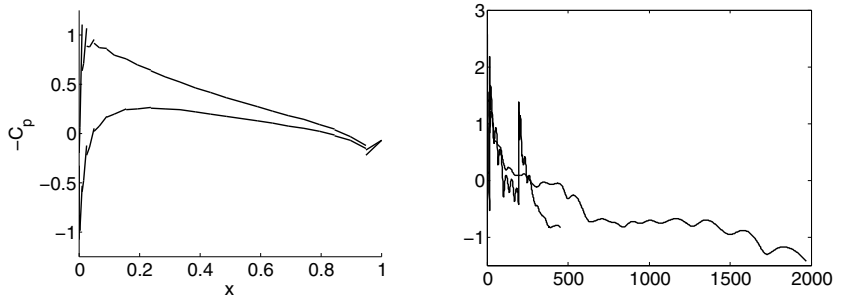

Fig. 2 MTC1 of the ADIGMA project. Left: Pressure coefficient, $p = 1$, $2\mathbb{D}$ Euler, NACA0012, $M = 0.5$, $\alpha = 2.0°$. Right: Mass conservation ("ρ") residual reduction without multigrid and with initial full FV multigrid followed by DG with agglomeration multigrid.

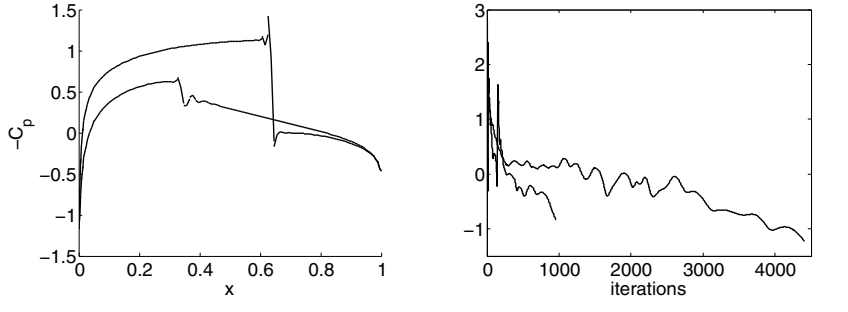

Fig. 3 MTC2 of the ADIGMA project. Left: Pressure coefficient, $p = 1$, $2\mathbb{D}$ Euler, NACA0012, $M = 0.8$, $\alpha = 1.25°$. Right: Mass conservation ("ρ") residual reduction without multigrid and with initial full FV multigrid followed by DG with agglomeration multigrid.

5 Conclusions

Agglomeration multigrid is a common strategy to accelerate convergence in vertex-centered finite-volume schemes. The implementation of a robust and effective agglomeration multigrid algorithm requires a substantial man-power investment. The availability of agglomeration multigrid in Edge was an important motivation to implement of our vertex-centered DG scheme inside Edge instead of attempting an implementation from scratch. To the best of our knowledge, this is the first time that such an extension effort has been attempted.

A central issue has been positively resolved: agglomeration multigrid, as developed for vertex-centered finite-volume schemes, is indeed effective as a converge acceleration also for the vertex-centered DG method. These first results certainly motivate further study of the method. First, the implementation needs to be completed to accept arbitrary number of mesh levels and orders $p > 1$. Also, for $p > 1$, it should be investigated whether a p-multigrid layer should be included before agglomoration multigrid is activated.

References

1. Berggren, M.: A vertex-centered, dual discontinuous Galerkin method. J. Comput. Appl. Math. 192(1), 175–181 (2006)
2. Berggren, M., Ekström, S.-E., Nordström, J.: A discontinuous Galerkin extension of the vertex-centered edge-based finite volume method. Commun. Comput. Phys. 5, 456–468 (2009)
3. Blazek, J.: Computational Fluid Dynamics, 2nd edn. Elsevier, Amsterdam (2005)
4. Ekström, S.-E., Berggren, M.: Incorporating a discontinuous Galerkin method into the existing vertex-centered edge-based finite volume solver Edge. In: Kroll, N., et al. (eds.) ADIGMA. NNFM, vol. 113, pp. 39–52. Springer, Heidelberg (2010)
5. Eliasson, P.: EDGE, a Navier–Stokes solver, for unstructured grids. Technical Report FOI-R-0298-SE, Swedish Defence Research Agency (2001)
6. FOI. Edge - Theoretical Formulation. Technical Report FOI dnr 03-2870, Swedish Defence Research Agency (2007) ISSN-1650-1942