Seventh International Conference on Computational Fluid Dynamics (ICCFD7), Big Island, Hawaii, July 9-13, 2012 ICCFD7-1606

Convergence Acceleration of High Order Numerical Simulations using a Hybrid Spectral Difference - Finite Volume Multigrid Method

Y. Allaneau[*,†], L. Y. Li[†] and A. Jameson[†]
Corresponding author: allaneau@stanford.edu

[*] Metacomp Technologies, Inc., CA 91301, USA
[†] Stanford University, CA 94305, USA.

Abstract: The goal of this paper is to show how numerical simulations of fluid flow using high order methods for unstructured meshes can be sped up using a hybrid multigrid method. In our work we accelerate the steady state convergence of a Spectral Difference code by coupling it to a Finite Volume solver. While we want to obtain the solution for the SD code, low frequency corrections to the solution are computed using the finite volume code.

Keywords: Numerical Algorithms, Computational Fluid Dynamics, Spectral Differences, Multigrid.

1 Introduction

High order numerical methods for unstructured meshes have recently received a lot of attention in the CFD community. However, these methods can be slow to converge due to the very strong limitations on the time steps that can be taken. While a lot of work is devoted in the community to developing implicit schemes to overcome this difficulty (see for example the work by Persson[1, 2] or Birken[3]), we focus instead on a multigrid approach to solving the problem.

1.1 Multigrid methods

Assume we are interested in solving a problem $R(u) = 0$ in a basis $\mathcal{B}_h$ counting n elements using an iterative method. Very often, the computation of the solution u_h in basis $\mathcal{B}_h$ to a good level of accuracy requires a very large number of iterations. The idea behind multigrid methods is to adjust an unconverged solution u_h by estimating a correction in a basis $\mathcal{B}_{2h}$ counting twice as less elements as basis $\mathcal{B}_h$. In the case of PDEs, the dimension of the solution space is decreased by coarsening the mesh (hence the notation $_h$ and $_{2h}$, where h denotes a characteristic length of the mesh). The physical interpretation to multigrid methods in this case is that the correction on the coarser mesh allows to capture large scale phenomenon more quickly. Our description of multigrid methods can be broadened however and projecting the problem to a basis counting less elements is just a way to speed up computations. Basically, we could estimate the corrections to the solution using a completely different method, even counting more degrees of freedom, as long as it allows us to obtain a rough estimate of the solution in a time quicker than the original method.

1.1.1 Linear problems

The multigrid method has been widely used to solve linear problems (Laplace equation for example). We describe here very briefly its working principle. Consider a linear problem $Lu = g$, where L is a linear operator and u the solution. Assume we can find an approximate solution u_h such that $Lu_h = g_h \neq g$ (we assume that L represent the same operator in the appropriate basis). There exists a correction v_h such that

$u = u_h + v_h$. Now since L is a linear operator, it follows that v_h must satisfy this equation:

$$Lv_h = g - g_h = e_h$$

While solving this problem exactly would be as expensive as the original one, the idea is instead to solve for $Lv_{2h} = e_{2h}$, where the subscript $_{2h}$ indicates that the vector representing the solution in a base $\mathcal{B}_{2h}$ has twice as less elements as one with a subscript $_h$ (by mesh coarsening for example). The original solution u_h is then corrected by the operation

$$u_h \leftarrow u_h + I_{2h}^{h} v_{2h}$$

It can be noted that a correction of v_{2h} could be obtained in a basis $\mathcal{B}_{4h}$ and the correction to the correction of v_{2h} could be computed in a basis $\mathcal{B}_{8h}$...

1.1.2 Euler equations

It was established very early that the multigrid method could accelerate the convergence transonic potential flow calculations[5, 6, 7, 8], although the governing equations are of mixed elliptic and hyperbolic type. Jameson[4] extended the approach to the Euler and Navier Stokes equations in the '80s. The problem is not as simple as when dealing with linear sets of equations. Indeed, an equation for the correction v_h is not immediately defined and instead we correct u_h using $v_{2h} = u_{2h} - I_{2h}^{h} u_h$, where u_{2h} is an estimate of the solution on a coarser mesh:

$$u_h \leftarrow u_h + I_{2h}^{h}\left(u_{2h} - I_{h}^{2h} u_h\right)$$

We define the Euler equations as $R(u) = 0$. As mentioned before we use an iterative method to solve for u (in this case we solve by homotopy, by introducing a pseudo time step). When we transfer the solution to the next coarser mesh to estimate the correction, we add a forcing term P_{2h} to the residual:

$$P_{2h} = I_{h}^{2h} R_h(u_h) - R_{2h}(u_{2h}^{(0)})$$

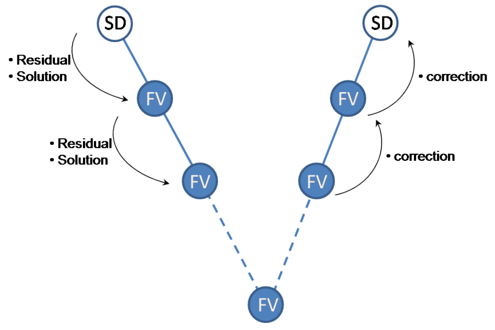
Therefore when we update the solution u_{2h} using a multistage time stepping scheme with the forcing term, the formula becomes

$$\begin{aligned}
u_{2h}^{(1)} &= u_{2h}^{(0)} - \alpha_0 \Delta t \left(R_{2h}(u_{2h}^{(0)}) + P_{2h}\right)\\
&\cdots\\
u_{2h}^{(q+1)} &= u_{2h}^{(0)} - \alpha_q \Delta t \left(R_{2h}(u_{2h}^{(q)}) + P_{2h}\right)\\
&\cdots\\
u_{2h}^{(k)} &= u_{2h}^{(0)} - \Delta t \left(R_{2h}(u_{2h}^{(k-1)}) + P_{2h}\right)\\
u_{2h}^{n+1} &= u_{2h}^{(k)}
\end{aligned}$$

Details on this scheme can be found in the original papers on the multigrid scheme by Jameson[4]. Two things should be noted. In the first stage of the scheme, P_{2h} cancels $R_{2h}(u_{2h}^{(0)})$ and replaces it with $I_{h}^{2h} R_h(u_h)$. Therefore, the solution is driven by the residual of the fine mesh. The other important thing is that when the solution is converged on the fine mesh, $R_h(u_h) = 0$. It follows that $u_{2h}^{(q+1)} = \cdots = u_{2h}^{(1)} = u_{2h}^{(0)} = I_{h}^{2h} u_h$. Therefore $v_{2h} = 0$ and the multigrid does not introduce unwanted corrections.

2 Hybrid multigrid method

Classical geometric multigrid methods for PDEs compute the solution on a fine mesh and corrections to the solution on successive coarser meshes to eliminate the various components of the error at a faster rate. The method relies on the fact that low frequency components of the error can be represented on coarser meshes and be damped or propagated outside of the numerical domain more efficiently on these meshes. In our work, the solution is computed on an unstructured mesh using the Spectral Difference method and low frequency corrections to the solution are performed on multiple levels using a Finite volume code. The reason behind this idea is that convergence acceleration using Finite Volume multigrid method is well known and a easier to implement than a full SD multigrid method.



2.1 SD and FV flow solvers

2.1.1 Spectral difference method

In this work, we used the Spectral Difference code developed in the Aerospace Computing Lab at Stanford. The code is based on the method pioneered by Kopriva *et al.* in 1996, as the 'conservative staggered-grid Chebyshev multi-domain method'[9]. It was then extended by Liu *et al.*[10], who first called it the spectral difference method, Wang *et al.*[11], who implemented it for simplex cells for 2D Euler equations, May *et al*[12], who applied it to 2D N-S equations, and Sun *et al.*[13], who implemented it for 3D hexahedral cells for 3D N-S equations. Premasuthan *et al.* then investigated and developed convergence acceleration techniques and shock capturing with artificial viscosity[14, 15, 16]. Jameson in 2010 gave a proof for energy stability of the SD method for linear PDEs for all orders of accuracy[18].

For an unstructured quadrilateral or hexahedral mesh, each element in the physical domain $\{x_i\}$ is firstly transformed into a standard element in the computational domain $\{\xi_i\} \in [0,1]^d$, where d is the dimension. The Euler equations in the physical domain is transformed into the computational domain as follows:

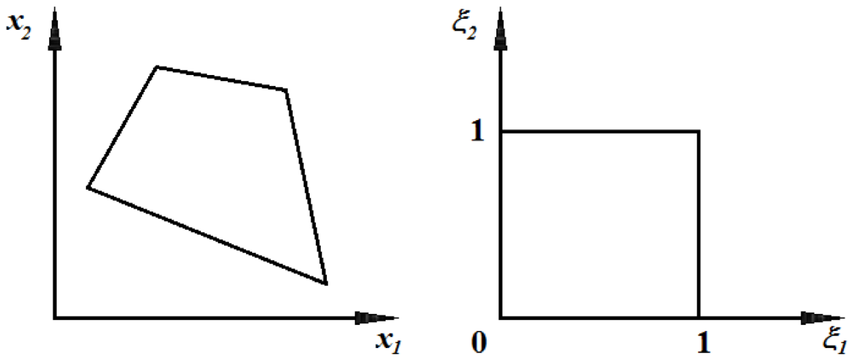


Figure 1: Transformation from (x_1, x_2) to (ξ_1, ξ_2).

$$\frac{\partial \tilde{w}}{\partial t} + \frac{\partial \tilde{F}_i}{\partial \xi_i} = 0 \tag{1}$$

where

$$\tilde{w} = |J| \cdot w, \quad \tilde{F}_i = |J| \cdot [J^{-1}]_{ij} F_j, \quad J = \frac{\partial \mathbf{x}}{\partial \xi} \tag{2}$$

The Basis Function and the Reconstructed Polynomials

Two staggered sets of points are defined in the standard element, namely the solution points and the flux points. The values of the conservative variables are calculated at the solution points, and the values of the flux vectors are computed at the flux points. In the following context, the conservative variables are also referred to as the solution variables. In each dimension, the number of solution points is one less than the number of flux points, such that the order of the reconstructed solution polynomial is one less than that of the reconstructed flux polynomial. Therefore the flux derivatives ($\partial\tilde{F}_i/\partial\xi_i$) are of the same order as the solution polynomial ($\tilde{w}$).

In each dimension, in order to reconstruct a solution polynomial of order $N-1$, N solution points and $N+1$ flux points are needed. The solution points (ξ_s)are defined to be at the Gauss-Lobatto points:

$$\xi_s = \frac{1}{2}\left[1-\cos\left(\frac{2s-1}{2N}\cdot\pi\right)\right], \quad s=1,2,...,N \tag{3}$$

and the flux points ($\xi_{s+\frac{1}{2}}$) at the Legendre-Gauss-Quadrature points to ensure stability of the scheme, as suggested by Huynh[17] and proved by Jameson[18].

The basis functions in 1D are the Lagrange polynomials $\{h_i\}_{i=1,...,N} \subset \mathcal{P}_{N-1}$ interpolated from the solution points, and $\{l_{i+\frac{1}{2}}\}_{i=0,...,N} \subset \mathcal{P}_N$ interpolated from the flux points, i.e.

$$\begin{aligned} h_i(\xi) &= \prod_{\substack{s=1\\ s\neq i}}^{N}\left(\frac{\xi-\xi_s}{\xi_i-\xi_s}\right) \\ l_{i+\frac{1}{2}}(\xi) &= \prod_{\substack{s=0\\ s\neq i}}^{N}\left(\frac{\xi-\xi_{s+\frac{1}{2}}}{\xi_{i+\frac{1}{2}}-\xi_{s+\frac{1}{2}}}\right) \end{aligned}$$

The reconstructed solution and flux polynomials in higher dimensions are therefore just the tensor products of these 1D polynomials.

Implementation in 2D

A standard element in 2D with solution points and flux points is illustrated in Figure 2. The reconstructed solution polynomial in the standard element is

$$w(\xi_1,\xi_2) = \sum_{j=1}^{N}\sum_{i=1}^{N}\frac{\tilde{w}_{i,j}}{|J_{i,j}|}\cdot h_i(\xi_1)\cdot h_j(\xi_2) \tag{4}$$

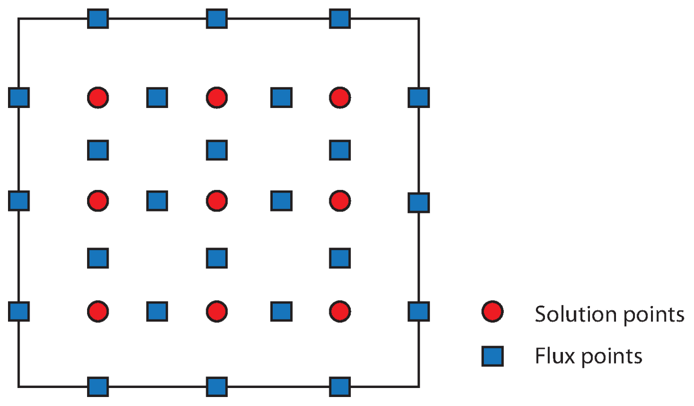


Figure 2: A SD standard element in 2D, with $N=3$.

Similarly, the reconstructed flux polynomials are

$$\tilde{F}_1 = \sum_{j=1}^{N}\sum_{i=0}^{N}\tilde{F}_{1_{i+\frac{1}{2},j}}\cdot l_{i+\frac{1}{2}}(\xi_1)\cdot h_j(\xi_2) \tag{5}$$

$$\tilde{F}_2 = \sum_{i=1}^{N}\sum_{j=0}^{N}\tilde{F}_{2_{i,j+\frac{1}{2}}}\cdot h_i(\xi_1)\cdot l_{j+\frac{1}{2}}(\xi_2) \tag{6}$$

The reconstructed solution polynomials are continuous within each element, but discontinuous at the element interfaces. Therefore an approximate Riemann solver is needed to compute a common flux at each interface flux points. In this work, a Rusanov solver is used. At each time step, the residual needed to update the solution vectors ($\partial\tilde{w}/\partial t$) is computed as follows:

1. Given the solution variables at the solution points, the solution variables are computed at the flux points using Equation (4).

2. The fluxes are computed at the interior flux points.

3. The fluxes at the interface flux points are computed using the Rusanov Riemann solver.

4. The flux derivatives at the solution points are calculated by differentiating Equation (5):

$$\left(\frac{\partial \tilde{F}_1}{\partial \xi_1}\right)_{i,j} = \sum_{j=1}^{N}\sum_{i=0}^{N} \tilde{F}_{1_{i+\frac{1}{2},j}} \cdot l'_{i+\frac{1}{2}}(\xi_1) \cdot h_j(\xi_2) \tag{7}$$

$$\left(\frac{\partial \tilde{F}_2}{\partial \xi_2}\right)_{i,j} = \sum_{i=1}^{N}\sum_{j=0}^{N} \tilde{F}_{2_{i,j+\frac{1}{2}}} \cdot h_i(\xi_1) \cdot l'_{j+\frac{1}{2}}(\xi_2) \tag{8}$$

5. Eventually, the residual at the solution points is

$$\left(\frac{\partial \tilde{w}}{\partial t}\right)_{i,j} = -\left(\frac{\partial \tilde{F}_1}{\partial \xi_1}\right)_{i,j} - \left(\frac{\partial \tilde{F}_2}{\partial \xi_2}\right)_{i,j} \tag{9}$$

2.1.2 Finite Volume solver

Our hybrid multigrid method is obtained by coupling the SD code to a preexisting multigrid finite volume code (based on FLO82 and FLO103). These 2D finite volume codes are designed to deal with O and C-meshes to compute inviscid and viscous flows around airfoils. They were developed in the '80s and are highly optimized for performance. Details on the implementation can be found in the same paper describing the multigrid[4]; this paper also contains the description of the JST artificial dissipation scheme used to stabilized simulations.

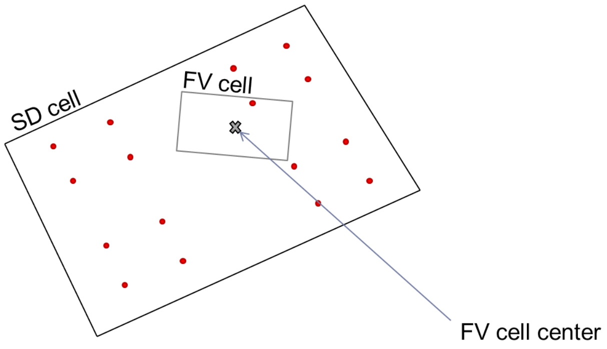
2.2 Transfers from SD to FV

The implementation of the multigrid method requires us to transfer both the solution u_{SD} and the residual $R_{SD}(u_{SD})$ to the finite volume code. We denote by $_{SD}$ quantities related to the SD mesh and by $_{FV}$ quantities related to the finest FV mesh. Also we denote by I_{SD}^{FV} the transfer operator from SD to FV and I_{FV}^{SD} the transfer operator from FV to SD. Note that the residual is actually transferred (operation $I_{SD}^{FV} R_{SD}(u_{SD})$) instead of computing the FV residual on the SD solution $R_{FV}(I_{SD}^{FV} u_{SD})$.

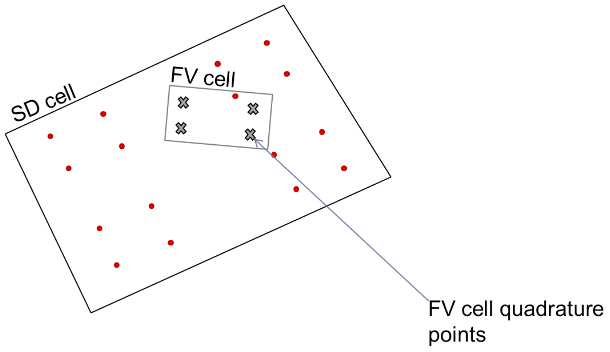
Transferring quantities from the SD mesh to a FV cell is relatively easy. The first most simple approach is to evaluate the SD solution (which is piecewise polynomial) at the FV cell center location. We then take the FV cell value to be the one of its center location. The second method is a slightly more complicated and estimates the average value of the finite volume cell by integration (numerical quadrature). Note that both these methods do not enforce conservation of the solution in time, which does not really matter as we are seeking for a steady state solution.

2.3 Transfers from FV to SD

Transferring the solution from the FV mesh to a SD node is not as easy and defining I_{FV}^{SD} requires more work. We start by isolating the SD node where we want to evaluate the quantity (step 1). Then we need to identify the neighboring FV cells to this node such that the dual cell (obtained by joining FV cell centers) contains the SD node (step 2). Once the FV dual cell has been identified, we need to evaluate the local coordinates (ξ, η) of the SD node inside this cell (step 3). Finally, we can evaluate the quantity at the SD node by bilinear reconstruction (step 4). Most of the work can be done at the beginning of the run and only step 4 is done during the actual multigrid cycle. The various steps are depicted in table 2.

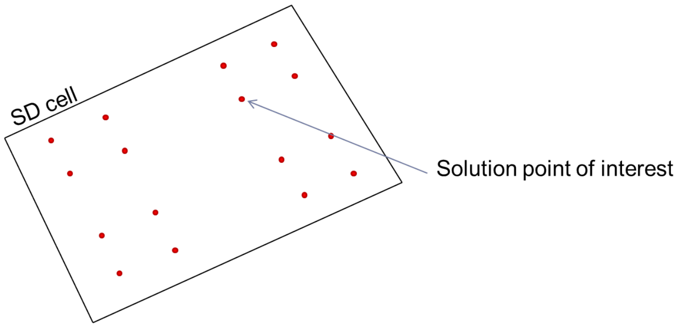


Method 1 - Compute the value at the center of the FV cell

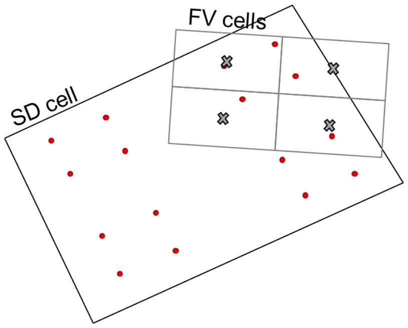


Method 2 - Integrate the values over the FV cell to estimate an average value

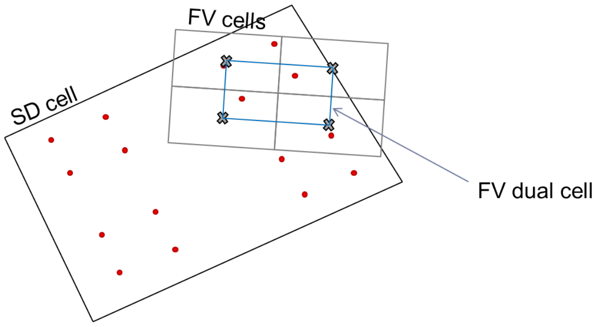
Table 1: Transfer of quantities from SD to FV



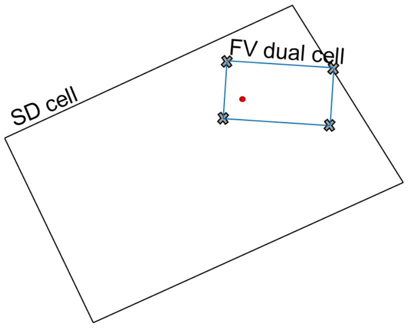
Step 1 - Identify the SD node of interest



Step 2 - Identify the FV cells around the node



Step 3 - Construct the dual FV cell and obtain the local coordinates (ξ, η) of the node in this cell

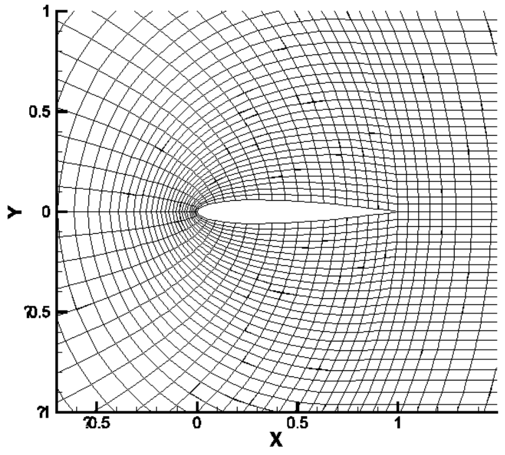


Step 4 - Evaluate the SD node value by bilinear interpolation inside the FV dual cell

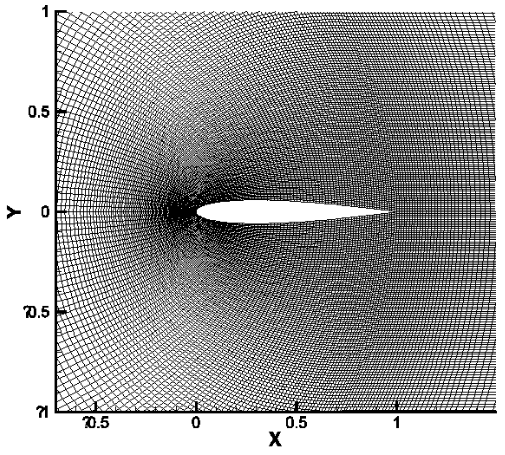
Table 2: Transfer of quantities from FV to SD

3 Results

The method was used to compute the steady state inviscid subsonic flow around a NACA 0012 airfoil. The solution is expected to be of order 4 on a mesh counting 2048 cells. The various finite volume meshes used in the multigrid cycles count 512×64 (32,768), 256×32 (8,192), 128×16 (2,048), 64×4 (256) and 32×2 (64) cells. Therefore, the multigrid counts a total of 6 levels in this example ($5 + 1$ for the SD mesh). We denote these meshes as $L_{SD}, L_1, L_2, L_3, L_4, L_5$, with L_1 being the finest FV mesh and L_5 the coarsest. We started by generating the fine Finite Volume mesh L_1 counting 512×64 cells. Then we considered 4×4 blocs of cells; by connecting the outside vertices, we created the SD quad mesh L_{SD}. The various coarser FV meshes were also obtained from the finest FV mesh, mesh L_k being obtained from L_{k-1} by picking every other nodes. As a consequence, L_{SD} and L_3 are the same meshes. Figure 3 depicts the SD mesh on the left and the fine FV mesh on the right in an area close to the airfoil. Figure 4 is a detail of meshes L_{SD} and L_1. The SD mesh is drawn in thick lines. On the right, we represent the SD solution points. In this case, we see 16 nodes per cell (4 in each direction) because we run the SD code at a 4^{th} order of accuracy.

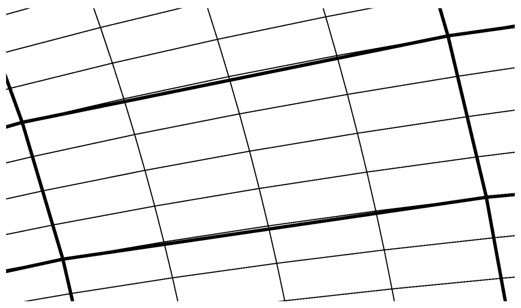


(a) SD mesh L_{SD}

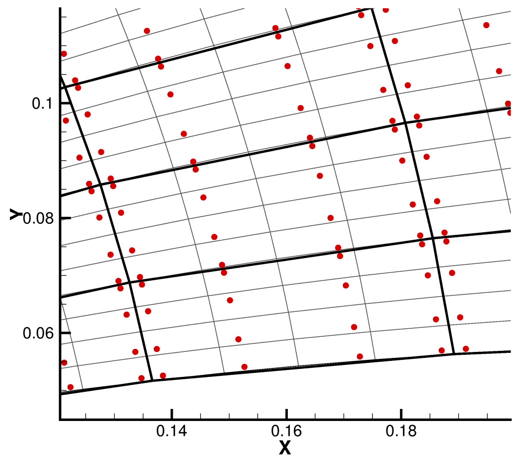


(b) FV fine mesh L_1

Figure 3: SD and fine FV meshes used for the computations



(a) Overlapping of L_{SD} and L_1



(b) SD solution points

Figure 4: Local comparison of SD and FV meshes

Figure 5 shows the flow solution around the airfoil using the multigrid method.

In Figure 6, the residual convergence is plotted. We notice a huge acceleration in convergence during the first steps (the residual is more than 2 orders of magnitude smaller for the hybrid multigrid method after 1000 steps). After that convergence seems to become the same for both method and the residuals keep

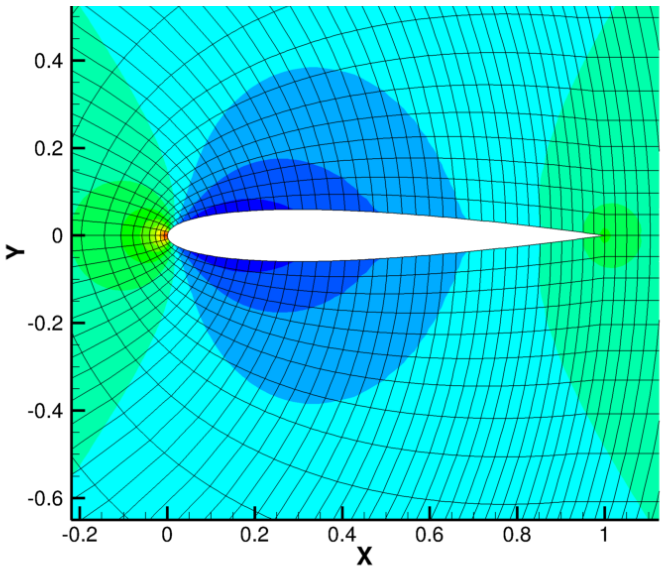
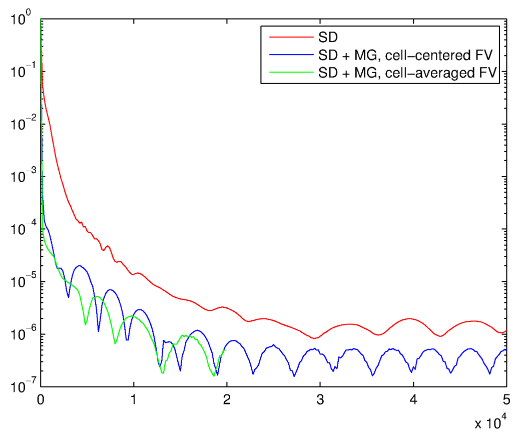
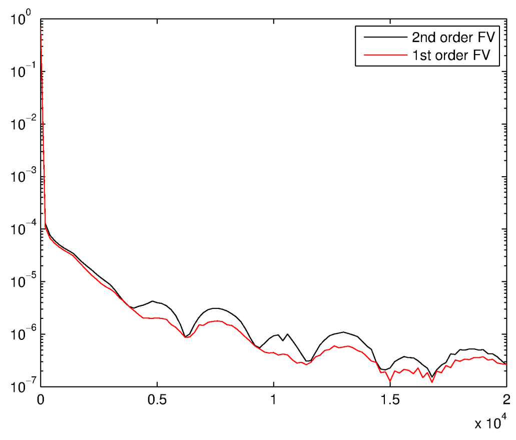


Figure 5: Density contour around the airfoil

decreasing at similar rates. At this point, we do not explain the slow down in residual convergence and we are still investigating this issue. For a pure finite volume scheme, the original multigrid method was able to converge to machine zero in a couple dozen iterations only. However, as can be seen on Figure 7, this initial speed up in convergence as dramatic effect on the convergence of other values of interest. In this case, we consider the same simulation at a 2° angle of attack. Note how fast we get an estimate of C_ℓ using the hybrid method.



(a) Convergence of Hybrid method (SD 4^{th} order)

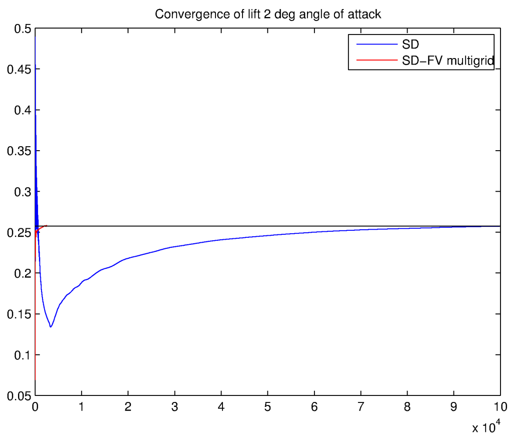


(b) Convergence of Hybrid method (SD 1^{st} and 2^{nd} order)

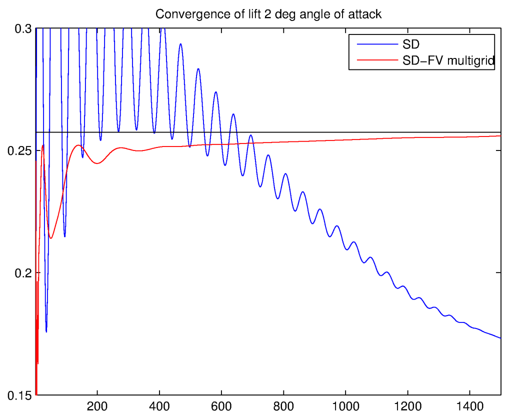
Figure 6: Residual convergence of SD and FV methods

4 Conclusion and Future Work

Considerable speed up of the SD method was obtained and preliminary results are very promising. Great accelerations are obtained both in the convergence of the residuals and in the convergence of quantities of interest such as the lift or drag. These improvements make the SD method (and by extension most of high order methods for unstructured meshes) a better candidate for real life computations. In the future,



(a) Global convergence



(b) Detail on first steps

Figure 7: Convergence of lift, 2° angle of attack

it will be interesting to compare speed up with implicit solvers. However, the use of high order method for unstructured meshes in more justified for the computation of unsteady flows and it would be interesting to see how this multigrid scheme can be used in conjonction to a dual pseudo time stepping scheme or as a preconditioner for an implicit solver.

Acknowledgments

The authors would like to thank Philipp Birken from the University of Kassel for his interesting suggestions as well as Guido Lodato from the Aerospace Computing Lab at Stanford. This work was supported in part by AFOSR Grant #FA 9550-07-1-0195 from the Computational Math Program under the direction of Dr. Fariba Fahroo and by NFS Grant #0915006 monitored by Dr. Leland Jameson. Yves Allaneau and Lala Yi Li were also supported by the Stanford Graduate Fellowship. The authors would also like to thank Metacomp Technologies for their support in this work.

References

[1] P.-O. Persson and J. Peraire, Newton-GMRES preconditioning for Discontinuous Galerkin Discretizations of the Navier-Stokes Equations, *Journal of Scientific Computing*, 30:2709-2733, 2008

[2] P.-O Persson, High-Order LES Simulations using Implicit-Explicit Runge-Kutta schemes, *AIAA paper* 2011-684, 2011

[3] P. Birken, G. Gassner, M. Hass and C.-D. Munz, A new class of preconditioners for discontinuous Galerkin methods for unsteady 3D Navier-Stokes equations: ROBO-SGS, *Journal of Computational Physics*, submitted.

[4] A. Jameson, Solution of the Euler Equations For Two Dimensional Transonic Flow by a Multigrid Method. *Applied Mathematics and Computations*, 13:327-356, 1983.

[5] J. C. South and A. Brandt, Application of Multi-Level Grid Method to Transonic Flow Calculations, *Hemisphere* 180-206, 1977

[6] A. Jameson, Acceleration of Transonic Potential Flow Calculations on Arbitrary Meshes by the Multiple Grid Method, *Proc AIAA 4th CFD conference*, 122-146, 1979

[7] D. R. McCarthy and T. A. Reyhner, Multigrid code for Three Dimensional Transonic Potential Flow About Inlets, *AIAA journal* 20:45-50, 1982

[8] D. A. Caughey, Multigrid Calculation of the Three Dimensional Transonic Potential Flows, *AIAA paper* 83-0374, 1983

[9] D.A. Kopriva, A Conservative Staggered-Grid Chebychev Multidomain Method for Compressible Flows. II. A Semi-Structured Method, *Journal of Computational Physics*, 128:475-488, 1996

[10] Y. Liu, M. Vinokur and Z.J. Wang, Spectral Difference Method for Unstructured Grids I: Basic Formulation, *Journal of Computational Physics*, 216:780-801, 2006.

[11] Z.J. Wang, Y. Liu, G. May and A. Jameson, Spectral Difference Method for Unstructured Grids II: Extension to the Euler Equations, *Journal of Scientific Computing*, 32:45-71, 2006

[12] G. May and A. Jameson, A Spectral Difference Method for the Euler and Navier Stokes Equations, *AIAA paper*, 2006-304, 2006

[13] Y. Sun, Z.J. Wang and Y. Liu, High-order Multidomain Spectral Difference Method for the Navier-Stokes Equations on Unstructured Hexahedral Grids, *Communications in Computational Physics*, 2:310-333, 2007

[14] S. Premasuthan, A Spectral Difference Method for Viscous Compressible Flows with Shocks, *AIAA paper*, 2009-3785, 2009

[15] S. Premasuthan, Computation of Flows with Shocks using Spectral Difference Scheme with Artificial Viscosity, *AIAA paper*, 2010-1449, 2010

[16] S. Premasuthan, Towards and Efficient and Robust High Order Accurate Flow Solver for Viscous Compressible Flow, *PhD thesis, Stanford University*, 2010

[17] H. T. Huynh, A Flux Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin Methods, *AIAA paper*, 2007-4079, 2007

[18] A. Jameson, A Proof of the Stability of the Spectral Difference Method for All Orders of Accuracy, *Journal of Scientific Computing*, 45:348-358, 2010