WEST-EAST HIGH SPEED FLOW FIELD CONFERENCE
19-22, November 2007
Moscow, Russia

IMPLICIT LU-SGS TIME INTEGRATION ALGORITHM FOR HIGH-ORDER SPECTRAL VOLUME METHOD WITH *p*-MULTIGRID STRATEGY

Matteo Parsani[*,1], **Kris Van den Abeele**[*,2] **and Chris Lacor**[*,3]

**Vrije Universiteit Brussel, Department of Mechanical Engineering, Fluid Dynamics and Thermodynamics Research Group*
Triomflaan 43, B-1050 Brussels, Belgium
[1]Email: mparsani@vub.ac.be, web page: http://mech.vub.ac.be/thermodynamics
[2]Email:kvdabeel@vub.ac.be, web page: http://mech.vub.ac.be/thermodynamics
[3]Email:chris.lacor@vub.ac.be, web page: http://mech.vub.ac.be/thermodynamics

Key words: Spectral Volume Method, LU-SGS Algorithm, *p*-Multigrid, Residual Restriction Operator.

Abstract. An efficient implicit lower-upper symmetric Gauss-Seidel (LU-SGS) solution algorithm has been combined for the first time with a high-order spectral volume method and a full *p*-Multigrid strategy to accelerate the convergence rate. The LU-SGS solver is preconditioned by the block element matrix, and the system of equations is the solved with an exact LU decomposition approach with pivoting per columns. The time integration methods considered are the backward Euler difference, the second- and third-order backwards differencing formulae and the fourth-order explicit-first-stage, single-diagonal-coefficient, diagonally-implicit Runge-Kutta method. The implicit solver with backwards schemes has shown a speed up factor of more than one order of magnitude relative to the well-known multi-stage optimized Runge-Kutta schemes for the quasi-1D Euler flow while the explicit-first-stage, single-diagonal-coefficient, diagonally-implicit Runge-Kutta method has demonstrated to be around four times more efficient then the optimized explicit Runge-Kutta smoothers. In addition, the efficiency of the *p*-Multigrid algorithm is investigated with two different residual restriction operators. The first one was proposed by K. Van den Abeele et al.[21] and it is well-defined for certain 1D problems, while the second one is presented in this paper and it is more general because it treats the residual which arise from the spatial discretization as a control-volume average quantity. It has shown that for the quasi-1D Euler flow and the optimized explicit Runge-Kutta smoothers the residual restriction operator proposed by K. Van den Abeele et al.[21] is more efficient than the general one. On the other hand, the two residual restriction operators do not show any differences in the convergence rates of the implicit LU-SGS solver.

1. INTRODUCTION

High-order spatial accurate numerical schemes are being developed for use in a variety of solution procedures. In computational fluid dynamics (CFD), high-order accurate schemes are being pursued for direct numerical, large eddy, computational aeroacoustic, turbulent combustion and biomedics simulations where accurate resolution of small scales requires large grid densities. In addition, since CFD is more and more used as an industrial design and analysis tool, the applications are often of industrial relevance, requiring unstructured grids for efficient meshing. High-order accuracy must therefore be achieved on such unstructured grids. Discontinuous Galerkin[1-3] (DG) schemes and the more recently developed Spectral Volume[4-7] (SV) and Spectral Difference[8-11] (SD) schemes are especially suited for these purposes[12-13]. The Discontinuous Galerkin method is based on the weighted residual form of the governing equations. The Spectral Volume method is based on the integral form of the governing equations while, the Spectral Difference method is based on the differential form.

However, when combined with classical solution methods, such as explicit Runge-Kutta (R-K) solvers, these schemes suffer from a restrictive CFL condition and hence a relatively low convergence, especially for viscous grids which are clustered in the viscous boundary layer. In fact, it is well-known that high-order methods are restricted to a smaller CFL number than low-order ones. Moreover, they possess much less numerical dissipation. Therefore it takes an excessive amount of CPU-time to reach a state-steady solution with explicit solvers. Hence, with high-order methods, it is possible to achieve low error levels more efficiently than with traditional first-order and second-order accurate schemes, but efficient implicit solution approaches are necessary to fully fulfill this potential.

Implicit time-integration schemes are highly suited to improve the efficiency of a solver, since they can advance the solution with significantly larger time steps compared to explicit methods. In Bijl et al.[14-15], implicit R-K solvers were investigated in combination with a standard cell-centered finite volume scheme with artificial dissipation added for stability. In these papers it is shown that Runge-Kutta schemes are susceptible to order reduction for some stiff systems. Moreover, it is concluded that significant potential improvements in the temporal efficiency of implicit schemes could be achieved from algebraic solver developments. In fact, one could speculate that solver improvements could be more dramatic than improvements in integration techniques.

In the present contribution an implicit lower-upper symmetric Gauss-Seidel (LU-SGS) approach is combined for the first time with the SV method and a full *p*-multigrid strategy[21] and it is shown to be a very efficient and stable solver. The LU-SGS algorithm dates from the late 80's. It was first formulated by Yoon and Jameson 1988 in the context of 2nd order central schemes and it was recently rediscovered by Wang and adapted for use with SV and SD schemes[19-20].

The paper is organized as follows. In the next section, the formulation of the spectral volume method is described. In Section 3 the full *p*-multigrid is presented. In Section 4, the LU-SGS approach is described for the backward Euler method and it is extended to higher-order temporal schemes such as 2nd, 3rd order backward differencing (BDF2 and BDF3) and the fourth-order, six-stages, explicit-first-stage, single-diagonal-coefficient, diagonally-implicit Runge-Kutta (ESDIRK64) method[14]. The numerical results for the steady state solution of the quasi-1D Euler equations in a convergent nozzle with a subsonic out-flow are presented in Section 5. Conclusions and future work~~s~~ are summarized in Section 6.

2. SPECTRAL VOLUME FORMULATION

The spectral volume (SV) method can be applied to hyperbolic conservation laws (1)

$$\frac{\partial \mathbf{Q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{Q}) = \mathbf{S}(\mathbf{Q}) \tag{1}$$

The computational domain V is divided in N_{SV} cells, called spectral volumes (SVs), with index i and volume V_i. Each of these SVs is further subdivided into control volumes (CV), with index j. Integrating (1) over such a CV and applying the Gauss theorem gives

$$\frac{\partial \overline{Q}_{i,j}}{\partial t} V_{i,j} = -\int_{\partial V_{i,j}} \mathbf{F} \cdot d\mathbf{s} + \int_{V_{i,j}} \mathbf{S} dV = R_{i,j} \tag{2}$$

where $V_{i,j}$ is the volume of CV j in cell i, $R_{i,j}$ is the residual and $Q_{i,j}$ is the CV average. On a spectral volume i, a polynomial approximation of the solution is defined

$$Q_i \approx q_i \equiv \sum_{j}^{N_i(p,d)} \overline{Q}_{i,j} \overline{L}_{i,j} \tag{3}$$

N_i (p, d) is the number of CVs in cell i, depending on the desired degree of the polynomial approximation p and the number of spatial dimensions d. The polynomials $\overline{L}_{i,j}$ associated to the CVs i,j are defined by

$$\frac{1}{V_{i,j}} \int_{V_{i,j}} \overline{L}_{i,m} dV = \delta_{jm} \tag{4}$$

where δ_{jm} is the Kronecker delta function. Equation (4) ensures the following property of the polynomial approximation

$$\frac{1}{V_{i,j}} \int_{V_{i,j}} q_i dV = \overline{Q}_{i,j} \tag{5}$$

With the polynomial approximation u_i, the flux integral and the source term integral in (2) can be approximated to order p + 1, using Gauss quadrature. On the boundary between two SVs however, there are two available values for the flux $\mathbf{F}$, one from within each SV. Thus on these boundaries a suitable Riemann flux $\mathbf{F}^R$, for instance the Lax-Friedrichs flux, must be used. A more elaborate overview of the SV method can be found in the work by Wang et al.[4-7].

3. P-MULTIGRID STRATEGY

The main idea of a multigrid algorithm is based on the observation that error-smoothing operators are generally very efficient in eliminating high-frequency errors, but much less adequate for the low-frequency errors. The multigrid strategy is to switch to a coarser representation of the solution, where the low-frequency errors on the fine representation occur as high-frequency modes, which can thus be efficiently damped out. In the traditional h-multigrid approach, this is done by switching to a coarser spatial grid. With a p-multigrid algorithm, a high-order solution representation is transferred to a lower-order one. Such p-multigrid algorithms have already been studied for high-order discontinuous Galerkin methods[23-26]. They have also been applied to spectral element methods, Ronquist and Patera[27]. Interesting properties such as p-independent convergence rates and locality of the transfer operators have been reported. The p-multigrid algorithm for a SV method described in this section is largely based on the algorithm used by Fidkowski et al.[25-26]. A simple, two-level Full Approximation Scheme algorithm as proposed by Brandt[28] can be summarized in the following way. To solve a fine level problem $\mathbf{R}^f(\overline{\mathbf{Q}}^f) = 0$, perform the following operations[21]:

- Perform $\boldsymbol{v_1}$ smoothing sweeps on the fine level: $\overline{\mathbf{Q}}^f \leftarrow (\mathbf{G}^f)^{v_1}\overline{\mathbf{Q}}^f$
- Transfer the state and the residual to the coarse level:

  $\overline{\mathbf{Q}}_o^c \leftarrow \widetilde{I}_f^c\overline{\mathbf{Q}}^f,\quad \mathbf{f}^c \leftarrow \mathbf{R}^c(\overline{\mathbf{Q}}^f) - \mathbf{R}^c(\overline{\mathbf{Q}}_0^c) = I_f^c\mathbf{R}^f(\overline{\mathbf{Q}}^f) - \mathbf{R}^c(\overline{\mathbf{Q}}_0^c)$
- Solve the coarse level problem: $\mathbf{R}^c(\overline{\mathbf{Q}}^c) = \mathbf{f}^c$
- Prolongate the coarse level error and correct the fine level state:

  $\overline{\mathbf{Q}}^f \leftarrow \overline{\mathbf{Q}}^f + I_c^f(\overline{\mathbf{Q}}^c - \overline{\mathbf{Q}}_0^c)$
- Perform $\boldsymbol{v_2}$ smoothing sweeps on the fine level: $\overline{\mathbf{Q}}^f \leftarrow (\mathbf{G}^f)^{v_2}\overline{\mathbf{Q}}^f$

In this algorithm, $\mathbf{G}^f$ represents an arbitrary smoothing operator on the fine level. $\mathbf{f}^c$ is the so-called forcing function. The coarse level problem could again be solved using a FAS algorithm, and so on. In this way, one arrives at a V-cycle. A further increase in efficiency can be achieved by initializing the solution on coarser levels. In this way, a better initial solution is provided for the fine levels, which will also improve the robustness of the method. This corresponds to a so-called Full Multgrid (FMG) algorithm. In the present work, the decision on when to start solving a finer level problem is made in an analogous way as in Fidkowski et al.[25-26]. The switch to a finer level is made when the L_2 norm of the coarse level residuals is smaller than a factor η times the L_2 norm of the fine level residual. Multigrid levels with $p = 0, p = 1$ and $p = 3$ are considered, corresponding to the following relation between p^{fine} and p^{coarse}: $(p^{fine}+1)/(p^{coarse}+1) = 2,$ i.e. doubling the order of accuracy with each higher multigrid level. This choice is made here because it implies that the resolved wave number range is doubled with each multigrid level as well, as is the case usually for h-multigrid algorithms. The high wave number range, which should be strongly damped, then corresponds to wave numbers K between $K_{\max}/2$ and $K_{\max}$, with $K_{\max}$ the maximum wave number which is resolved on the considered multigrid level.

The prolongation (I_f^c), state restriction $(\widetilde{I}_f^c)$ and residual restriction (I_c^f) operators still have to be defined for the SV method. In Kris Van den Abeele et al.[21] these operators are chosen as follows (with the omission of the SV index *i*):

- *Prolongation operator*: this operator can be chosen in the same way as for discontinuous Galerkin methods. On the coarse as well as on the fine level, the solution within a SV is represented by a polynomial $\sum_{j=1}^{N(p,d)}\overline{Q}_jL_j$. The coarse level polynomials L_j^c can be written as a function of the fine level polynomials L_m^f:

$$L_j^c = \sum_{m=1}^{N_f}\alpha_{jm}L_m^f,\quad j,m = 1,\dots,N_c \tag{6}$$

N_c and N_f are the number of CVs within a SV on the coarse and fine level. By equating the fine level solution to the coarse level solution, the following expression for I_c^f : $(I_c^f)_{mj} \equiv \alpha_{jm}$.

- *State restriction operator*: this operator can also be defined completely analogously as for discontinuous Galerkin methods, by projecting the fine level solution onto the coarse polynomial basis. This results in the following definition for $\widetilde{I}_f^c$: $(\widetilde{I}_f^c)_{mj} = (P^{-1}Q)_{mj}$. The matrices **P** and **Q** are defined by

$$P_{jm} = \int_V L_j^c L_m^c dV, \quad j, m = 1, \ldots, N_c \tag{7}$$

$$Q_{jm} = \int_V L_j^c L_m^f dV, \quad j = 1, \ldots, N_c, \qquad m = 1, \ldots, N_f \tag{8}$$

- *Residual restriction operator*: in this paper we consider two residual restriction operators. The first one (VdA RRO) was proposed by K. Van den Abeele et al.[21] and it is well-defined for certain 1D problems. In this approach the forcing term of a fine level CV is divided between the coarse level CVs which share part of their domain with it and each coarse level CV receives a part proportionate to the fraction of the fine level CV they occupy. However, in that work, only multigrid levels $p = 0, p = 1$ and $p = 3$ are used. The distribution of CVs inside a 1D SV is then such that two fine level CVs, with index $j1$ and $j2$, correspond exactly to one coarse level CV with index j. The restricted residual is then calculated as

$$\left[I_f^c \mathbf{R}^f \left(\overline{\mathbf{Q}}^f \right) \right]_j \equiv R_j^c \left(\overline{\mathbf{Q}}^f \right) = R_{j1}^f \left(\overline{\mathbf{Q}}^f \right) + R_{j2}^f \left(\overline{\mathbf{Q}}^f \right) \tag{9}$$

The CV distribution for the different p-multigrid levels and the residual restriction operation are illustrated in Fig. 1, where d indicate the local coordinate of the most external CV boundaries.

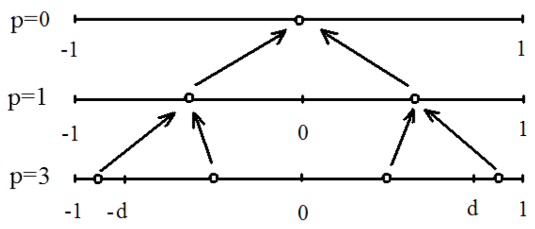

Figure 1. Distribution of CVs within a 1D SV for a first-, second- and fourth-order method and illustration of the residual restriction process.

In the present paper this residual restriction operator is used to compare the convergence rate to the steady state solution of the implicit LU-SGS algorithm with the convergence rate of the explicit Runge-Kutta smoothers used in K. Van den Abeele et al.[21].

Moreover, in this work a new general residual restriction operator is introduced (GRRO). It is based on the idea that the residuals (after dividing them by the CV volume) are also CV averaged quantities, just like the solution, so in fact the residual restriction operator can be:

$$(I_f^c)_{jk} = V_j^c (\widetilde{I}_f^c)_{jk} \frac{1}{V_k^f} \tag{10}$$

This operator is more general than the one that was used in K. Van den Abeele et al.[21], and is valid for any 2D or 3D cell as well. In Section 5 the results obtained with both residual restriction operators and both the R-K and LU-SGS smoothers will be compared.

4. IMPLICIT LU-SGS ALGORITHM

Consider the integration of the system of ordinary differential equations (ODEs) represented by the equation

$$\frac{\partial \mathbf{Q}}{\partial t} - \mathbf{R}(\mathbf{Q}(t)) = 0 \tag{11}$$

In the present case, the vector **Q** results from the semi-discretization of the governing equations of fluid mechanics. The responsibility of the integrator is to integrate any **R** with which it is provided. Trouble often arises when the Jacobian of **R**, $(\partial \mathbf{R}/\partial \mathbf{Q})$, has a broad range of eigenvalues. This may give rise to stiffness. Nontrivial near-wall stiffness is common in practical engineering problems due to grid clustering that increases with Reynolds number.

Popular implicit ODE integration methods are defined at both extremes by multistep or multistage methods. Whole classes of schemes, however, combine both approaches in an attempt to correct the flaws of either the multistep or multistage methods as the Modified Extended Backward Differencing Formulae.

Implicit multistep BDF methods compute each **Q-**vector update to design order of accuracy using one nonlinear set of equations to be solved at each time-step. Unfortunately, they are not absolutely stable (A-stable) above second-order. Additionally, they are not self-starting and have diminished properties when used in a variable step-size context. Practical experience indicates that large-scale engineering computations are seldom stable if run with BDF4. The BDF3 scheme, with its smaller regions of instability, is often stable but diverges for certain problems and some spatial operators[14-15].

On the other hand, multistage Runge-Kutta schemes are self-starting, are easily implemented in a variable time-stepping mode, and can be designed with A-stability properties for any temporal order. In general, higher-order schemes in both space and time become more attractive as the need for accuracy increases, i.e. error tolerances decrease. Consequently, multistage R-K schemes are potentially an efficient alternative to second-order methods (of which the second-order backward formula (BDF2) which is L-stable is the most popular) for problems requiring high accuracy, such as direct turbulent simulations and long-range wave propagation. However, these schemes require multiple nonlinear solves at each time-step. Practical R-K methods are the explicit- first-stage, single-diagonal-coefficient, diagonally-implicit R-K methods which are characterized by a lower triangular form of the coefficient table, thus resulting in a single implicit solve at each individual stage[14-15].

The efficiency of an implicit time-marching method that involves subiterations depends on the iterative technique used to solve the nonlinear problem or problems arising at each time step. In the last one and a half decades, many implicit algorithms have been developed and applied successfully to unstructured grids to accelerate convergence to steady state[16-18]. In this paper the numerical results obtained with a new combination of an LU-SGS algorithm and a full p-multigrid strategy are presented.

4.1. Backward Euler difference

Starting from eq. (11), the backward Euler difference for the spectral volume sv can be written as

$$\frac{\mathbf{Q}_{sv}^{n+1}-\mathbf{Q}_{sv}^{n}}{\Delta t}-\left[\mathbf{R}_{sv}(\mathbf{Q}^{n+1})-\mathbf{R}_{sv}(\mathbf{Q}^{n})\right]=\mathbf{R}_{sv}(\mathbf{Q}^{n}) \tag{12}$$

Let $\Delta\mathbf{Q}_{sv}=\mathbf{Q}_{sv}^{n+1}-\mathbf{Q}_{sv}^{n}$. After linearizing the residual, we obtain

$$\mathbf{R}_{sv}(\mathbf{Q}^{n+1})-\mathbf{R}_{sv}(\mathbf{Q}^{n})\approx\frac{\partial\mathbf{R}_{sv}}{\partial\mathbf{Q}_{sv}}\Delta\mathbf{Q}_{sv}+\sum_{nb\neq c}\frac{\partial\mathbf{R}_{sv}}{\partial\mathbf{Q}_{nb}}\Delta\mathbf{Q}_{nb} \tag{13}$$

where nb indicates all the neighboring cells contributing to the residual of cell sv. Therefore, the fully linearized equations for (12) can be written as

$$\left(\frac{\mathbf{I}}{\Delta t}-\frac{\partial\mathbf{R}_{sv}}{\partial\mathbf{Q}_{sv}}\right)\Delta\mathbf{Q}_{sv}-\sum_{nb\neq sv}\frac{\partial\mathbf{R}_{sv}}{\partial\mathbf{Q}_{nb}}\Delta\mathbf{Q}_{nb}=\mathbf{R}_{sv}(\mathbf{Q}^{n}) \tag{14}$$

However, it takes lot of memory to store the LHS implicit Jacobian matrices. Therefore, we employ a LU-SGS scheme to solve (14), i.e., we use the most recent solution for the nb cells,

$$\left(\frac{\mathbf{I}}{\Delta t}-\frac{\partial\mathbf{R}_{sv}}{\partial\mathbf{Q}_{sv}}\right)\Delta\mathbf{Q}_{sv}^{(k+1)}=\mathbf{R}_{sv}(\mathbf{Q}^{n})+\sum_{nb\neq sv}\frac{\partial\mathbf{R}_{sv}}{\partial\mathbf{Q}_{nb}}\Delta\mathbf{Q}_{nb}^{*} \tag{15}$$

This results in a block diagonal system, which is solved cell-wise with a direct LU decomposition solver. Since we do not want to store the $\left(\partial\mathbf{R}_c/\partial\mathbf{Q}_{nb}\right)$, (15) is further manipulated as follows:

$$\mathbf{R}_{sv}(\mathbf{Q}^{n})+\sum_{nb\neq sv}\frac{\partial\mathbf{R}_{sv}}{\partial\mathbf{Q}_{nb}}\Delta\mathbf{Q}_{nb}^{*}=\mathbf{R}_{sv}(\mathbf{Q}^{n},\{\mathbf{Q}_{nb}^{n}\})+\sum_{nb\neq sv}\frac{\partial\mathbf{R}_{sv}}{\partial\mathbf{Q}_{nb}}\Delta\mathbf{Q}_{nb}^{*}\approx\mathbf{R}_{sv}(\mathbf{Q}^{n},\{\mathbf{Q}_{nb}^{*}\}) \tag{16}$$

Hence

$$\mathbf{R}_{sv}(\mathbf{Q}^{n},\{\mathbf{Q}_{nb}^{*}\})\approx\mathbf{R}_{sv}(\mathbf{Q}^{*},\{\mathbf{Q}_{nb}^{*}\})-\frac{\partial\mathbf{R}_{sv}}{\partial\mathbf{Q}_{sv}}\Delta\mathbf{Q}_{sv}^{*}=\mathbf{R}_{sv}(\mathbf{Q}^{*})-\frac{\partial\mathbf{R}_{sv}}{\partial\mathbf{Q}_{sv}}\Delta\mathbf{Q}_{sv}^{*} \tag{17}$$

But,

$$\Delta \mathbf{Q}_{sv}^{*} = \mathbf{Q}_{sv}^{n+1,(k)} - \mathbf{Q}_{sv}^{n} \equiv \Delta \mathbf{Q}_{sv}^{(k)} \tag{18}$$

Let $\Delta \widetilde{\mathbf{Q}}_{sv}^{(k+1)} = \Delta \mathbf{Q}_{sv}^{(k+1)} - \Delta \mathbf{Q}_{sv}^{(k)} = \mathbf{Q}_{sv}^{n+1,(k+1)} - \mathbf{Q}_{sv}^{n+1,(k)}$. Then combining (15), (17) and (18) and adding and subtracting $(\mathbf{I}/\Delta t)\Delta Q_{sv}^{(k)}$ we obtain

$$\left(\frac{\mathbf{I}}{\Delta t} - \frac{\partial \mathbf{R}_{sv}}{\partial \mathbf{Q}_{sv}}\right)\Delta \widetilde{\mathbf{Q}}_{sv}^{(k+1)} = \mathbf{R}_{sv}(\mathbf{Q}^{*}) - \frac{\mathbf{I}}{\Delta t}\Delta \mathbf{Q}_{sv}^{(k)} \tag{19}$$

4.2. 2nd-order backward differencing (BDF2)

In the same way, introducing the BDF2 method for the time derivative, i.e.,

$$\mathbf{Q}^{n+1} = \frac{4}{3}\mathbf{Q}^{n} - \frac{1}{3}\mathbf{Q}^{n-1} + \frac{2}{3}\Delta t \quad \mathbf{R}(\mathbf{Q}^{n+1}) \tag{20}$$

and following the same steps described for the backward Euler difference, we obtain

$$\left(\frac{\mathbf{I}}{\Delta t} - \frac{2}{3}\frac{\partial \mathbf{R}_{sv}}{\partial \mathbf{Q}_{sv}}\right)\Delta \widetilde{\mathbf{Q}}_{sv}^{(k+1)} = \frac{2}{3}\mathbf{R}_{sv}(\mathbf{Q}^{*}) + \frac{1}{3}\mathbf{Q}_{sv}^{n}\frac{\mathbf{I}}{\Delta t} - \frac{1}{3}\mathbf{Q}_{sv}^{n-1}\frac{\mathbf{I}}{\Delta t} - \frac{\mathbf{I}}{\Delta t}\Delta \mathbf{Q}_{sv}^{(k)} \tag{21}$$

4.3. 3rd-order backward differencing (BDF3)

Discretizing the time derivative in equation (11) with 3rd-order backward differencing, i.e.,

$$\mathbf{Q}^{n+1} = \frac{11}{8}\mathbf{Q}^{n} - \frac{9}{11}\mathbf{Q}^{n-1} + \frac{2}{11}\mathbf{Q}^{n-2} + \frac{6}{11}\Delta t \quad \mathbf{R}(\mathbf{Q}^{n+1}) \tag{22}$$

and following the same procedure used for the previous schemes we find

$$\begin{aligned}\left(\frac{\mathbf{I}}{\Delta t} - \frac{6}{11}\frac{\partial \mathbf{R}_{sv}}{\partial \mathbf{Q}_{sv}}\right)\Delta \widetilde{\mathbf{Q}}_{sv}^{(k+1)} &= \frac{6}{11}\mathbf{R}_{sv}(\mathbf{Q}^{*}) + \frac{7}{11}\mathbf{Q}_{sv}^{n}\frac{\mathbf{I}}{\Delta t} - \frac{9}{11}\mathbf{Q}_{sv}^{n-1}\frac{\mathbf{I}}{\Delta t} + \\ &\quad + \frac{2}{11}\mathbf{Q}_{sv}^{n-2}\frac{\mathbf{I}}{\Delta t} - \frac{\mathbf{I}}{\Delta t}\Delta \mathbf{Q}_{sv}^{(k)}\end{aligned} \tag{23}$$

4.4. 4th-order explicit-first-stage, single-diagonal-coefficient, diagonally-implicit Runge-Kutta method

A general ESDIRK scheme of s stages is given by the following[14]:

$$\frac{\mathbf{Q}_{sv}^{p} - \mathbf{Q}_{sv}^{n}}{\Delta t} - \sum_{j=1}^{p} a_{pj}\mathbf{R}_{sv}(\mathbf{Q}^{j}) = 0 \tag{24}$$

$$\mathbf{Q}_{sv}^{n+1} = \mathbf{Q}_{sv}^{n} + \Delta t\sum_{j=1}^{s} b_{j}\mathbf{R}_{sv}(\mathbf{Q}^{j}) = 0 \tag{25}$$

where $p = 1,....,s$, a_{pj} and b_j are the coefficients of the scheme which can be found in the Butcher tableau (see Bijl et al.[14]).

This scheme is characterized by a lower triangular form of the coefficient table, thus resulting in a single implicit solve at each individual stage. The first stage is explicit ($a_{p1} = 0$), i.e.

$$\mathbf{Q}_c^{p=1} = \mathbf{Q}_c^n \tag{26}$$

and the last stage coefficient takes on the form $a_{pj} = b_j$, thus enabling eq. (27) to simplified as

$$\mathbf{Q}_{sv}^{n+1} = \mathbf{Q}_{sv}^{p=s} \tag{27}$$

Hence, eq. (24) is only used for $p = 2,....,s-1$.

Applying the procedure described for the BDF methods to eq. (24), leads the general equation for the stage *p*

$$\left(\frac{\mathbf{I}}{\Delta t} - a_{pp}\frac{\partial \mathbf{R}_{sv}}{\partial \mathbf{Q}_{sv}}\right)\Delta\widetilde{\mathbf{Q}}_{sv}^{p,(k+1)} = a_{pp}\mathbf{R}_{sv}(\mathbf{Q}^*) + \sum_{j=1}^{p} a_{pj}\mathbf{R}_{sv}(\mathbf{Q}^j) - \frac{\mathbf{I}}{\Delta t}\Delta\mathbf{Q}_{sv}^{p,(k)} \tag{28}$$

where *k* is the index of the Symmetric Gauss-Seidel sweep.

In this paper the well-known ESDIRK64, i.e. 4th-order six-stages, is used (see Appendix in Bijl et al.[14]).

4.5 Iterative technique

Equations (19), (21), (23) and (28) are then solved with multiple symmetric forward and backward sweeps with a prescribed convergence tolerance ε. At each sweep the linear system is solved with an LU factorization with column pivoting. Note that if these equations are solved to machine zero, the unsteady residual is zero at each time step. The initial guess for $\mathbf{Q}_{sv}^{n+1}$ can be set to $\mathbf{Q}_{sv}^{n+1}$. Therefore, the initial "unsteady residual" is the same as the steady residual at the last time step, i.e., $\mathbf{R}_{sv}\left(\mathbf{Q}^n\right)$. The unsteady residual is then monitored for convergence. For steady state problems, it is not necessary to drive the unsteady residual to machine zero. In fact, it may be more efficient to set a maximum number of sweeps.

5. NUMERICAL RESULTS

In this section numerical results for the Steady quasi-1D Euler subsonic flow through a convergent nozzle are presented. This test case was first used in Wang[4]. Considering the Ideal Gas Model, the quasi-1D Euler equations are given by

$$\frac{\partial}{\partial t}\begin{Bmatrix}\rho A\\ \rho uA\\ \rho EA\end{Bmatrix}+\frac{\partial}{\partial x}\begin{Bmatrix}\rho uA\\ \left(\rho u^2+p\right)A\\ \left(\rho uE+up\right)A\end{Bmatrix}=\begin{Bmatrix}0\\ p\dfrac{\partial A}{\partial x}\\ 0\end{Bmatrix} \tag{29}$$

where the energy per unit mass E is defined by

$$E=\frac{1}{\gamma-1}\frac{p}{\rho}+\frac{1}{2}u^2 \tag{30}$$

$A=A(x)$ is the area of the convergent nozzle and it is given by

$$\begin{cases}A(x)=\dfrac{3}{2}-\dfrac{1}{2}\tanh(x)\\ -5\le x\le 5\end{cases} \tag{31}$$

The inflow and outflow conditions are:

$$\begin{aligned}&\rho_{in}=1.2849245,\quad u_{in}=0.30891936,\quad p_{in}=1.025685\\ &\rho_{out}=1.0,\quad u_{out}=0.8\frac{m}{s},\quad p_{out}=0.71428571\end{aligned} \tag{32}$$

The processor used is an Intel(R) Core(TM)2 CPU T7200 2.00 GHz processor.

5.1. Optimized explicit Runge-Kutta schemes

In K. Van den Abeele et al.[21] the steady state solution was computed with the 4th-order SV4D08U-scheme combined with a Flux Difference Splitting (FDS) Riemann flux, on a uniform mesh with 100 cells. A full p-multigrid algorithm combined with multi-stage optimized explicit N_{RK}-stages Runge-Kutta smoothers were used[21-22], starting from an initial solution which varies linearly between the inlet and the outlet conditions. To ensure an efficient damping, a local time stepping technique was implemented. In K. Van den Abeele et al.[21] the multi-stage explicit Runge-Kutta smoothers of the following form were considered:

$$\begin{aligned}&\overline{Q}_{ij}^{0}=\overline{Q}_{ij}^{n}\\ &\overline{Q}_{ij}^{m}=C_m^1\overline{Q}_{ij}^{0}+C_m^2\overline{Q}_{ij}^{m-1}+C_m^3\frac{\Delta t}{\left|V_{ij}\right|}R_{ij}^{m-1},\qquad 1\le m\le N_{RK}\\ &\overline{Q}_{ij}^{n+1}=\overline{Q}_{ij}^{N_{RK}}\end{aligned} \tag{33}$$

and the coefficients that define the optimized explicit Runge-Kutta smoothers are listed in Table 1.

Scheme	C_m^1	C_m^2	C_1^3	C_2^3	C_3^3	C_4^3	C_5^3
Opt RK2	1	0	1/4	1	-	-	-
Opt RK3	1	0	1/5	1/2	1	-	-
Opt RK5	m=1: 0 m≠1: 1- C_m^3	m=1: 0 m≠1: C_m^3	85/1300	1/10	9/50	1/4	132/300

Table 1. Coefficients of the different optimized explicit Runge-Kutta schemes.

For the full *p*-multigrid calculation 1st-, 2nd- and 4th-order in space was used. The Runge-Kutta smoothers and the parameters of these computations are summarized in Table 2, where v_1 and v_2 are the number of sweeps in the descending and the ascending part of a V-cycle (Figure 2).

Algorithm	***p*-MG level**	**Spatial scheme**	**Smoother**	**CFL(σ)**	v_1	v_2
FMG	3	SV4D08U	Opt RK5	0.1	2	2
FMG	2	SV2U	Opt RK3	0.65	3	3
FMG	1	SV1U	Opt RK2	1.20	16	-
SG	-	SV4D08U	Opt RK5	0.4	1	-

Table 2. Computation parameters of the quasi-1D Euler equations with explicit optimized Runge-Kutta schemes in combination with the spatial scheme they are used with.

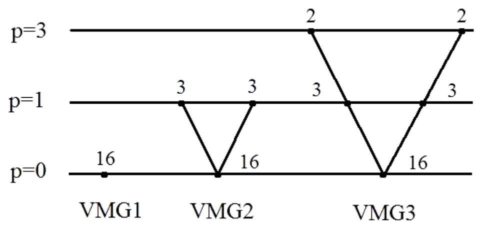

Figure 2. Used V-cycles, with number of sweeps on each multigrid level.

Figure 3 shows the residual history with the optimized explicit Runge-Kutta schemes for a single grid (a) and for full *p*-multigrid calculation (b).

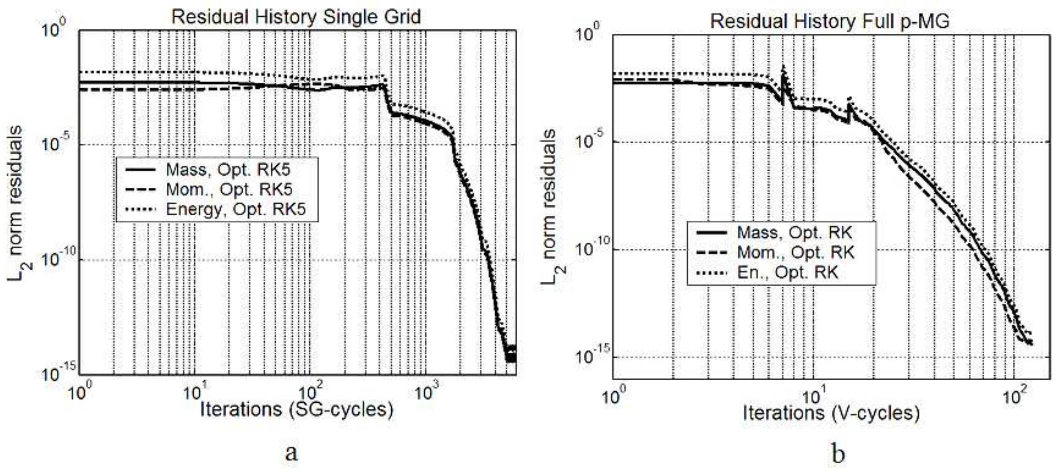

Figure 3. Residuals history of the quasi-1D Euler flow computations; optimized RK5 and single grid calculation (a), optimized R-K schemes and full *p*-multigrid calculation (b); 4th-order spatial accuracy.

The measured CPU-times per iteration are listed in Table 3, while in Table 4 the estimated total CPU-times for the residual L_2 norm to drop 11 orders of magnitude are indicated.

	SG (p=3)	**VMG1**	**VMG2**	**VMG3**
Opt RK	0.58 s	1.45 s	3.05 s	9.47 s

Table 3. Measured CPU-times ([s]) for the quasi-1D Euler flow computations, on a mesh with 100 cells, for the optimized R-K schemes.

Calculation	**SG**	**VMG1**	**VMG2**	**VMG3**	**Total CPU-time**
SG, Opt RK	4801	-	-	-	≈ 2784 s
FPMG, Opt RK	-	7	8	86	≈ 811 s

Table 4. Estimated total CPU-times ([s]) for the residual L_2 norm to drop 11 orders of magnitude, for quasi-1D Euler flow computations, on a mesh with 100 cells with the optimized R-K schemes.

5.2. Implicit LU-SGS algorithm

The aim of this section is to compare the efficiency (measured in CPU-time) of the LU-SGS algorithm for the backward Euler difference, BDF2, BDF3 and ESDIRK64 schemes with the efficiency of the optimized explicit Runge-Kutta smoothers described in the previous section.

The steady state solution was computed with the 4th-order SV4D08U-scheme combined with a Flux Difference Splitting (FDS) Riemann flux, on a uniform mesh with 100 cells. The LU-SGS algorithm combined with backward Euler difference, BDF2, BDF3 and ESDIRK64 was used for the single grid computations, while for the full p-multigrid strategy the LU-SGS algorithm was combined only with backward Euler difference and ESDIRK64 because BDF schemes above first-order are not self-starting. Actually, these properties do not allow to implement BDF methods above second-order with the p-multigrid algorithm in an easy and efficient way. In fact, BDF2 and BDF3 are multistep schemes and they need the solution at one or two previous time-levels to advance the solution in time. That means that at each p-multigrid level it is necessary to transfer the solution at the previous time-levels too.

In general, the convergence rate of the simulation is affected by several parameters. Obviously the *CFL* number is an important convergence parameter. In the present study, the *CFL* number is computed based on the following power law form

$$CFL = C_1 \cdot C_2^{\,n} \tag{34}$$

where C_1 and C_2 are two constants which depend on the order of magnitude of the residual~~s~~ of the energy equation and on the implicit scheme; n is the iteration number.

Moreover, the efficiency of a time-marching method that involves subiterations depends on the mechanism chosen to terminate the subiterations. This can be based on a fixed number of subiterations, on a prescribed convergence tolerance, or some other criterion. Whatever criterion is selected, a parameter is introduced which must be chosen to maximize the efficiency. Consequently, to maximize the efficiency it is necessary to carefully consider the setting of every parameter.

The values of the constants C_1 and C_2 and the number of the LU-SGS sweeps to maximize the efficiency are listed in Table 5. In this table, two values are indicated for the two constants: the first one is used when the residual~~s~~ of the energy equation is greater than 10^{-4} while the second one is used when the same quantity is lower than 10^{-4}.

Computation	***C_1***	***C_2***	**LU-SGS sweeps**
LU-SGS BE	5, 100	1.5, 5	3
LU-SGS BDF2	5, 100	1.35, 5	3
LU-SGS BDF3	5, 100	1.3, 5	3
LU-SGS ESDIRK64	3, 100	1.15, 5	1

Table 5. Parameters used for the CFL number for the quasi-1D Euler flow calculations with single grid strategy and the implicit LU-SGS solver.

In Figure 4 the residuals histories for a single grid strategy obtained with the optimized RK5 scheme are compared with those obtained with backward Euler, BDF2, BDF3 and ESDIRK64 schemes with LU-SGS solver.

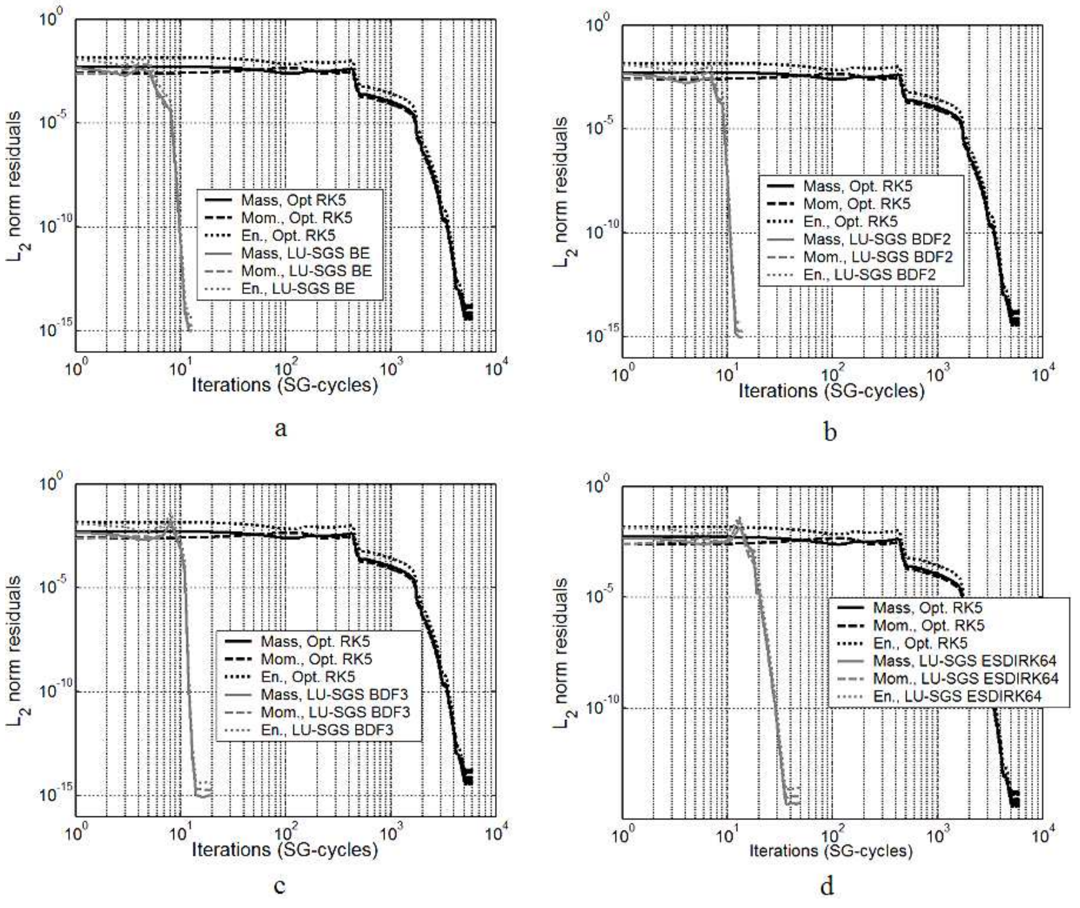

Figure 4. Residuals histories comparison of the quasi-1D Euler flow computations for backward Euler difference (a), BDF2 (b), BDF3 (c) and ESDIRK64 (d); implicit LU-SGS solver and optimized RK5 smoother; single grid calculations; 4th-order spatial accuracy.

It is interesting to note (Table 5) that for the ESDIRK64 only one LU-SGS sweep is sufficient. If only one LU-SGS sweep had been set for the backward Euler, BDF2 and BDF3 schemes, the calculations, with the same parameters, would have blown up. The other way around, if the CFL law had been changed the efficiency would have decreased drastically.

The number of the single grid iterations and the measured CPU-times for the residual to drop 11 orders of magnitude are listed Table 6.

Computation	SG Iterations	Time per iteration	Total CPU-time
LU-SGS BE	11	7.21 s	≈ 79 s
LU-SGS BDF2	13	7.51 s	≈ 98 s
LU-SGS BDF3	14	7.90 s	≈ 111 s
LU-SGS ESDIRK64	41	16.56 s	≈ 596 s

Table 6. Estimated total CPU-times ([s]) for the residual L_2 norm to drop 11 orders of magnitude, for the quasi-1D Euler flow computations, on a mesh with 100 cells for backward Euler difference, BDF2, BDF3 and ESDIRK64 with LU-SGS solver; single grid calculations; 4th-order spatial accuracy.

From Tables 4 and 6 it is seen that the backward Euler method with the LU-SGS solver is the most efficient. In fact, it is ~ 35 times faster than the optimized RK5, while BDF2, BDF3 and ESDIRK64 are respectively ~ 28, ~ 25 and ~ 4.5 times faster than the optimized smoother.

In Table 7, 8, 9 the parameters used for the full *p*-multigrid calculations are listed. They have been chosen to obtain the maximum efficiency for the total CPU-time. VMG1, VMG2, VMG3 indicate the V-cycles (Figure 2) and the numbers in each row correspond to the number of sweeps on each *p*-multigrid level.

Computation	VMG1 cycles	C_1	C_2	SGS sweeps
LU-SGS BE	3	15	1	3
LU-SGS ESDIRK64	2	15	1	1

Table 7. Parameters used for the full *p*-multigrid calculations in VMG1; quasi-1D Euler flow computations, on a mesh with 100 cells for backward Euler difference and ESDIRK64 with LU-SGS solver.

Computation	VMG2 cycles	C_1	C_2	SGS sweeps
LU-SGS BE	2, 3, 1	10	3	3
LU-SGS ESDIRK64	2, 2, 1	3	2	1

Table 8. Parameters used for the full *p*-multigrid calculations in VMG2; quasi-1D Euler flow computations, on a mesh with 100 cells for backward Euler difference and ESDIRK64 with LU-SGS solver.

Computation	VMG3 cycles	C_1	C_2	SGS sweeps
LU-SGS BE	1, 2, 3, 1, 1	10	10	3
LU-SGS ESDIRK64	1, 2, 2, 1, 1	5	3	1

Table 9. Parameters used for the full *p*-multigrid calculations in VMG3; quasi-1D Euler flow computation, on a mesh with 100 cells for backward Euler difference and ESDIRK64 with LU-SGS solver.

In Figure 5 the residuals histories with the optimized explicit RK schemes are compared with those obtained with implicit LU-SGS with backward Euler and ESDIRK64 schemes with full *p*-multigrid algorithm.

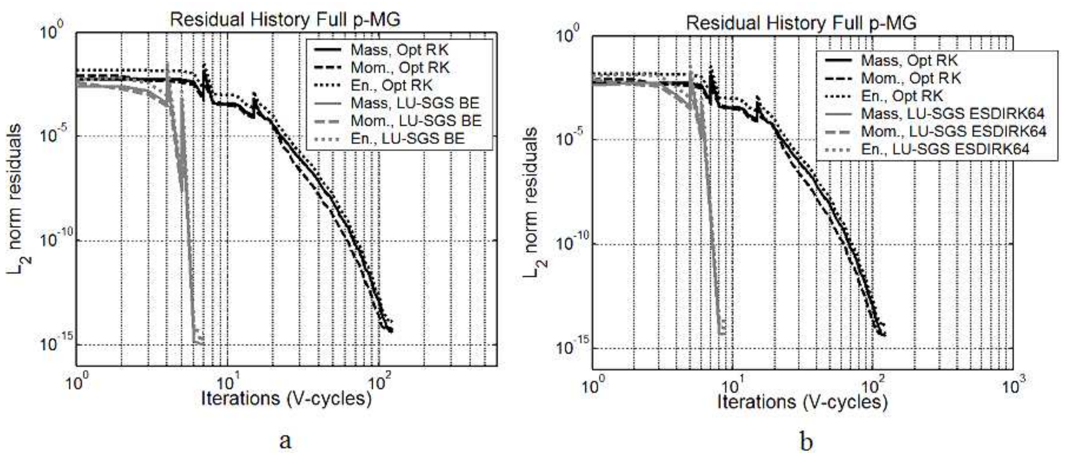

Figure 5. Residuals histories comparison of the quasi-1D Euler flow computations for backward Euler difference (a) and ESDIRK64 (b); full *p*-multigrid calculations.

The number of VMG1, VMG2 and VMG3 cycles for the residual to drop 11 orders of magnitude are listed Table 10, while the times take by each cycles are indicated in Table 11. Finally, the measured CPU-times for the residual to drop 11 orders of magnitude are listed Table 12.

Computation	**n^o VMG1 cycles**	**n^o VMG2 cycles**	**n^o VMG3 cycles**
LU-SGS BE	4	1	1
LU-SGS ESDIRK64	5	1	2

Table 10. Number of VMG1, VMG2 and VMG3 cycles for the residual to drop 11 orders of magnitude; quasi-1D Euler flow computations, on a mesh with 100 cells for backward Euler difference and ESDIRK64 with LU-SGS solver; full *p*-multigrid calculations.

Computation	**Time VMG1 cycle**	**Time VMG2 cycle**	**Time VMG3 cycle**
LU-SGS BE	3.09 s	9.59 s	24.84 s
LU-SGS ESDIRK64	6.98 s	25 s	60.07 s

Table 11. Time ([s]) takes by each VMG1, VMG2 and VMG3 cycle for quasi-1D Euler flow computations, on a mesh with 100 cells for backward Euler difference and ESDIRK64 with LU-SGS solver; full *p*-multigrid calculations.

Computation	**Total CPU-time**
LU-SGS BE	≈ 47 s
LU-SGS ESDIRK64	≈ 180 s

Table 12. Estimated total CPU-times ([s]) for the residual L_2 norm to drop 11 orders of magnitude for the quasi-1D Euler flow computations, on a mesh with 100 cells; backward Euler difference and ESDIRK64 with LU-SGS solver; full *p*-multigrid calculations.

From Tables 4 and 12 it is seen that the backward Euler method with the implicit LU-SGS solver and *p*-multigrid strategy is still the most efficient. In fact it is ~ 17 times faster then the optimized smoothers, while ESDIRK64 is 4.5 times faster than the optimized RK schemes.

As mentioned in Section 3, the same full *p*-multigrid calculations have been repeated using the more general definition of the residual restriction operator (9). First of all, a comparison between with the same smoothers is made. The number of each V-cycle and the measured CPU-times for the residual to drop 11 orders of magnitude are listed Tables 13 and 14 and 15.

Computation	**n° VMG1 cycles**	**n° VMG2 cycles**	**n° VMG3 cycles**
Optimized RK	7	8	162
LU-SGS BE	4	1	1
LU-SGS ESDIRK64	5	1	2

Table 13. Number of VMG1, VMG2 and VMG3 cycles for the residual to drop 11 orders of magnitude; quasi-1D Euler flow computations, on a mesh with 100 cells with residual restriction operator (10); optimized RK smoothers, backward Euler difference and ESDIRK64 with LU-SGS solver; full *p*-multigrid calculations.

Computation	**Time VMG1 cycle**	**Time VMG2 cycle**	**Time VMG3 cycle**
Optimized RK	1.45 s	2.93 s	9.01 s
LU-SGS BE	3.09 s	9.53 s	24.74 s
LU-SGS ESDIRK64	6.98 s	24.7 s	59.91 s

Table 14. Time ([s]) takes by each VMG1, VMG2 and VMG3 cycle for quasi-1D Euler flow computations, on a mesh with 100 cells with residual restriction operator (10); optimized RK smoothers, backward Euler difference and ESDIRK64 with LU-SGS solver; full *p*-multigrid calculations.

Computation	**Total CPU-time**
Optimized RK	≈ 1493 s
LU-SGS BE	≈ 71 s
LU-SGS ESDIRK64	≈ 179 s

Table 15. Estimated total CPU-times ([s]) for the residual L_2 norm to drop 11 orders of magnitude for the quasi-1D Euler flow computations, on a mesh with 100 cells with residual restriction operator (10); optimized RK smoothers, backward Euler difference and ESDIRK64 with LU-SGS solver; full *p*-multigrid calculations.

A comparison between the two residual restriction operators shows that for the LU-SGS solver the residual restriction operator used in K. Van den Abeele et al.[21] (eq. (9)) is efficient as the one defined by equation (10). However, from Tables 13 and 15 it is seen that for the optimized RK smoothers the efficiency decrease drastically. In fact, their convergence rate with the residual restriction operator (10) is ~ 1.8 slower than the previous one.

Figure 6 shows the residual histories for the optimized explicit RK smoothers. The residuals histories of the LU-SGS solver with backward Euler and ESDIRK64 methods with the two different residual restriction operators are not showed because the lines are at the top of each others.

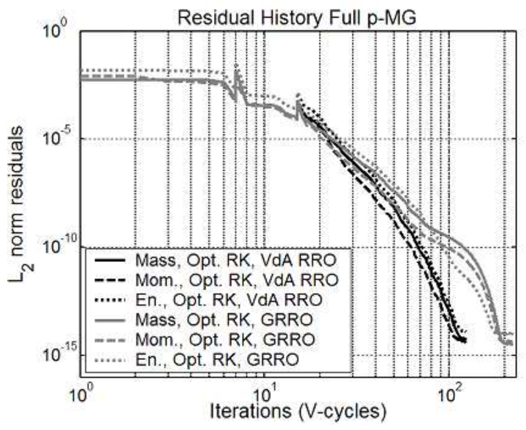

Figure 6. Residuals histories comparison of the quasi-1D Euler flow computations for the optimized RK smoothers; full *p*-multigrid calculations with the residual restriction operator (9) (VdA RRO) and the residual restriction operator (10) (GRRO).

It is interesting to note (Figure 6) that for both residual restriction operators the residual histories are exactly the same for VMG1 and VMG2 V-cycles, i.e. for 1st- and 2nd-order. In fact, for 1st- and 2nd-order, equation (10) is equivalent to the approach used by K. Van den Abeele et al.[21].

To make a fair comparison between the efficiency of the *p*-multigrid strategy with the two residual restriction operators the solution of the quasi-1D Euler flow was computed on finer meshes. For this purpose two meshes with respectively 400 and 800 cells were used. However, it was not possible to show any differences because each calculation takes the same number of VMG3 cycles for the residual L_2 norm to drop 11 orders of magnitude.

6. CONCLUSIONS

In this paper, an efficient implicit lower-upper symmetric Gauss-Seidel (LU-SGS) solution algorithm with full *p*-multigrid strategy and on a single grid calculation have been developed for high order spectral volume method. The implicit scheme with backward Euler difference and optimized parameters has shown more than an order of magnitude of speed-up relative to the multi-stage optimized Runge-Kutta explicit time integration schemes for the quasi-1D Euler flow. Moreover, it has demonstrated to be also more efficient with the well-known 4th-order explicit-first-stage, single-diagonal-coefficient, diagonally-implicit Runge-Kutta method.

In addition, the efficiency comparison between two different residual restriction operator has been made. It is shown that for the steady quasi-1D Euler flow the general residual restriction operator is less efficient than one used in K. Van den Abeele et al.[21] if optimized RK smoothers are used. On the other hand, the two residual restriction operators do not show any differences in the convergence rates of the implicit LU-SGS solver.

We are currently extending the approach to the general 2D viscous flow model on unstructured hexahedral grids with the two residual restriction operators and the results will be presented in the future publications.

7. REFERENCES

[1] B. Cockburn, and C.W. Shu, “TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: General framework”, *Mathematics of Computation* 52 (1989) 411.

[2] B. Cockburn, S.Y. Lin and C.W. Shu, “TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One-dimensional systems’’, *Journal of Computational Physics* 84 (1989) 90.

[3] B. Cockburn, S. Hou and C.W. Shu, “TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: The multidimensional case”, *Mathematics of Computation* 54 (1990) 545.

[4] Wang Z.J. and Liu Yen, “Spectral (Finite) Volume Method for Conservation Laws on Unstructured Grids III: Extension to One-Dimensional Systems”, *Journal of Scientific Computing* 20 (1) (2004) 137-157.

[5] Wang Z.J., Zhang L. and Liu Yen, “Spectral (Finite) Volume Method for Conservation Laws on Unstructured Grids IV: extension to two-dimensional systems”, *Journal of Computational Physics* 194 (2004) 716-741.

[6] Liu Y., Vinokur M. and Wang, Z.J., “Spectral (Finite) Volume Method for Conservation Laws on Unstructured Grids V: Extension to Three-Dimensional Systems”, *Journal of Computational Physics* 212 (2006) 454-472.

[7] Sun Yuzhi, Wang Z.J. and Liu Yen, “Spectral (Finite) Volume Method for Conservation Laws on Unstructured Grids VI: Extension to Viscous Flow”, *Journal of Computational Physics* 215 (2006) 41-58.

[8] Liu Y., Vinokur M. and Wang Z.J., “Discontinuous Spectral Difference Method for Conservation Laws on Unstructured Grids”, *3rd International Conference in CFD*, Toronto, Canada (2004).

[9] Liu Yen, Vinokur M. and Wang Z.J., Multi-Dimensional Spectral Difference Method for Unstructured Grids, AIAA-2005-0320.

[10] Wang Z.J. and Liu Yen, “The Spectral Difference Method for the 2D Euler Equations on Unstructured Grids”, AIAA-2005-5112.

[11] Huang P.G., Wang Z.J. and Liu Yen, “An Implicit Space-Time Spectral Difference Method for Discontinuity Capturing Using Adaptive Polynomials”, AIAA-2005-5255.

[12] Sun Y. and Wang Z.J., “Evaluation of discontinuous Galerkin and spectral volume methods for scalar and system conservation laws on unstructured grids”, *International Journal for Numerical Methods in Fluids* 45 (2004) 819-838.

[13] M. Zhang and C.W. Shu, “An analysis of and a comparison between the discontinuous Galerkin and the spectral finite volume methods”, *Computers and Fluids* 34 (2005) 581-592.

[14] H. Bijl, M. H. Carpenter and V. N. Vatsa, “Time Integration Schemes for the Unsteady Navier-Stokes Equations”, AIAA-2612-2001.

[15] H. Bijl, M. H. Carpenter, V. N. Vatsa and C. A. Kennedy, “Implicit Time Integration Schemes for the Unsteady Compressible Navier-Stokes Equations: Laminar Flow”, *Journal of Computational Physics* 179 (2002) 313-329.

[16] Venkatakrishnan V. and Mavriplis D.J., “Implicit Solvers for Unstructured Meshes”, *Journal of Computational Physics* 105 (1) (1993) 83-91.

[17] Venkatakrishnan V. and Mavriplis D.J., “Implicit Method for the Computation of Unsteady Flows on Unstructured Grids”, *Journal of Computational Physics* 127 (2) 380-397.

[18] Blanco M. and Zingg D.W., “A Fast Solver for the Euler Equation on Unstructured Grids Using a Newton-GMRES Method”, AIAA-97-0331.

[19] R.F. Chen and Z.J. Wang, “Fast, Block Lower-Upper Symmetric Gauss-Seidel Scheme for Arbitrary Grids”, *AIAA Journal* 38 (12) (2000).

[20] Yuzhi Sun and Z.J. Wang, “Efficient Implicit LU-SGS Algorithm for High-Order Spectral Difference Method on Unstructured Hexahedral Grids”, *AIAA*-2007-313.

[21] Kris Van den Abeele, Tim Broeckhoven and Lacor Chris, “Dispersion and dissipation properties of the 1d spectral volume method and application to a *p*-multigrid algorithm”, *Journal of Computational Physics* 224 (2) (2006) 616-636.

[22] C. Lacor, S. Smirnov and M. Baelmans, "A finite volume formulation of compact central schemes on arbitrary structured grids", *Journal of Computational Physics* 198 (2004) 535–566.

[23] B.T. Helenbrook, D.J. Mavriplis and H.A. Atkins, "Analysis of *p*-multigrid for continuous and discontinuous finite element discretizations", AIAA-2003-3989.

[24] F. Bassi and S. Rebay, "Numerical solution of the Euler equations with a multi-order discontinuous finite element method", *Second International Conference on Computational Fluid Dynamics*, Sydney, Australia (2002).

[25] Krzysztof J. Fidkowski, "A high-order discontinuous Galerkin multigrid solver for aerodynamic applications", Master's Thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, June 2004.

[26] Krysztof J. Fidkowski, Todd A. Oliver, James Lu and David L. Darmofal, "*p*-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations", *Journal of Computational Physics* 207 (2005) 92–113.

[27] E.M. Ronquist and A.T. Patera, "Spectral element multigrid I. Formulation and numerical results", *Journal of Scientific Computing* 2 (4) (1987) 389406.

[28] A. Brandt, Guide to Multigrid Development, Springer, Berlin (1982).