EVALUATION OF IMPLICIT TIME MARCHING SCHEMES FOR HIGH-ORDER SPECTRAL DIFFERENCE METHODS

Eduardo Jourdan, eduardojourdan92@gmail.com[1]
Fábio Mallaco Moreira, fabiom91@gmail.com[1]
Carlos Breviglieri, carbrevi@gmail.com[1]
André R. B. Aguiar, ufabc.andre@gmail.com[1]
João Luiz F. Azevedo, joaoluiz.azevedo@gmail.com[2]

[1]*Instituto Tecnológico de Aeronáutica, 12228-900 São José dos Campos, SP, Brazil*
[2]*Instituto de Aeronáutica e Espaço, 12228-904 São José dos Campos, SP, Brazil*

Abstract: *The present work is concerned with assessing different implicit schemes that could provide convergence acceleration for the spectral difference (SD) method when applied to steady state test cases. The study evaluates issues associated with computational performance in terms of convergence rate, CPU time and memory usage. The methods here considered are the generalized minimum residual (GMRES) algorithm and the lower-upper factorization symmetric Gauss-Seidel (LU-SGS) method. Different preconditioners for the GMRES algorithm are tested, such as the incomplete lower-upper factorization (ILU) with various levels of fill-in. Convergence is also analyzed with respect to some parameters of these methods, for instance, Krylov subspace size for the GMRES algorithm and number of sweeps in the LU-SGS method. The results provide data for discussions regarding performance, stability and robustness for implicit time-marching schemes. In particular, GMRES algorithm good convergence rate is highlighted as well as its large memory use as the method order of accuracy increases.*
Keywords: *GMRES, Implicit time scheme, High-order methods, Spectral difference methods, CFD*

1. INTRODUCTION

High-order numerical schemes represent the natural extension of current computational fluid dynamics (CFD) methods, which were developed over the past thirty years for aerospace simulations. The current generation methods are mostly 2nd-order accurate and have achieved a level of maturity and robustness desirable for everyday deployment in aeronautical engineering scenarios. Likewise, several complementary methods were developed for time integration, convergence acceleration, shock capturing and for dealing with geometric complexities. However, there are many problems that cannot be fully simulated using low-order methods, such as vortex dominated flows. Moreover, high-order methods offer the possibility to reduce simulation costs for given solution accuracy levels, when compared to low-order schemes.

There is room for improvement in many areas for high-order methods that must be pursued before they can compete with industrial-level CFD solvers. Computational resource requirements and run time are typical metrics used to classify a specific method or a combination of methods in a CFD solver. High-order schemes must cope with implicit schemes, limiters or filters and mesh manipulation techniques that also need to be superior, in comparison with the low-order counterparts. Therefore, a high-order method coupled with a low-order mesh with linear elements, for instance, will degrade the accuracy of the method near the domain boundaries. In order to overcome this problem and to fully realize the advantages of high-order methods, a correct description of curved boundaries is mandatory. Explicit methods for time integration suffer yet a stronger limitation of the time step value allowed for stability. Furthermore, even current implicit schemes (May *et al.*, 2010) are typically not robust enough and present unsatisfactory convergence rates for high-order spatial discretization.

The interest in the Spectral Difference (SD) method was prompted by the limitations encountered by the research group with other high order methods, namely the Spectral Finite Volume (SFV) and the Weighted Essentially Non-Oscillatory (WENO) schemes. The SFV method is restricted to triangular meshes, in 2-D, which limit its capability to capture boundary layer effects. The WENO schemes, on the other hand, present a very high computational cost and are difficult to efficiently implement in parallel. These limitations are expected to be overcome with the SD method since it is based on quadrilateral grids and have simple and straightforward implementation.

The present work main objective is to assess several already available convergence accelerators with the SD scheme (Moreira *et al.*, 2016) for spatial discretization and evaluate performance in terms of convergence rate, CPU time and memory usage. The methods here considered are the generalized minimum residual (GMRES) algorithm (Saad and Schultz, 1986) and lower-upper factorization symmetric Gauss-Seidel (LU-SGS) method (Sun *et al.*, 2007a). Different preconditioners for the GMRES algorithm are tested, such as incomplete lower-upper factorization (ILU) with various levels of fill-in. Convergence is also analyzed according to some parameters of those methods: Krylov subspace size for the GMRES algorithm and number of sweeps in the LU-SGS method.

The present draft is organized as follows, Section 2presents the numerical formulation of the high-order SD scheme. Section 3discusses the different implicit-time marching schemes used. Section 4presents the numerical results and compares them in terms of performance with literature results. Finally, Section 5draws concluding remarks and discusses ongoing efforts and suggestions for improvements in the context of the present work.

2. SPECTRAL DIFFERENCE METHOD FORMULATION

The flows of interest in the present work are assumed to be adequately modeled the 2-D Euler equations. The Euler equations describe the most general flow configuration for a non-viscous, non-heat conducting fluid. They are formed by the combination of three conservation laws, namely the conservation of mass, the momentum equations and conservation of energy, that are combined with the equation of state for perfect gases to form a closed system. These equations can be written in differential form as

$$\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} = 0 \,. \tag{1}$$

The vector of conserved variables, Q, and the convective flux vectors, E and F, are given by

$$Q = \left\{ \begin{array}{c} \rho \\ \rho u \\ \rho v \\ \varepsilon \end{array} \right\} , \qquad E = \left\{ \begin{array}{c} \rho u \\ \rho u^2 + p \\ \rho uv \\ (\varepsilon + p)u \end{array} \right\} , \qquad F = \left\{ \begin{array}{c} \rho v \\ \rho uv \\ \rho v^2 + p \\ (\varepsilon + p)v \end{array} \right\} . \tag{2}$$

The aforementioned equation of state for perfect gases can be written as

$$p = (\gamma - 1) \left[\varepsilon - \frac{1}{2} \rho (u^2 + v^2) \right] , \tag{3}$$

where ε is the total energy per unit volume and the ratio of specific heats, γ, is set as 1.4 for all computations in this work.

The Spectral Difference (SD) method employs a finite difference-like scheme and, to achieve an efficient implementation, all elements in the physical domain, (x, y), are transformed into a unit square element in the computational domain. Such transformation can be written as

$$\left(\begin{array}{c} x \\ y \end{array} \right) = \sum_{s=1}^{K} M_s(\xi, \eta) \left(\begin{array}{c} x_s \\ y_s \end{array} \right) , \tag{4}$$

where K is the number of points used to define the physical element, (x_s, y_s) are the Cartesian coordinates of such points, and $M_s(\xi, \eta)$ are the shape functions of the geometric transformation. In the case of a 1st-order linear boundary mesh, the transformation is bilinear and the analytic expression can be easily found. However, in the case of higher order meshes, required to represent accurately curved geometries, the number of points used to define a single cell increases. Considering a bi-polynomial representation, the transformation parameters can be calculated by numerically solving a linear system of size K.

The metric terms and the Jacobian matrix of the transformation can be computed in a pre-processing step and kept in memory given the stationary aspect of the mesh for the problems here considered. The implementation follows the formulation presented in Wang *et al.* (2007) and May and Jameson (2006). The governing equations in the physical domain are transformed into the computation domain, and are rewritten as

$$\frac{\partial \widetilde{Q}}{\partial t} + \frac{\partial \widetilde{E}}{\partial \xi} + \frac{\partial \widetilde{F}}{\partial \eta} = 0 \,, \tag{5}$$

where $\widetilde{Q} = |J| \, Q$ and J is the Jacobian matrix of the coordinate transformation, given by

$$J = \left(\begin{array}{cc} x_\xi & x_\eta \\ y_\xi & y_\eta \end{array} \right) . \tag{6}$$

For the current implementation, the flux vectors in the computational domain can be simplified from the general form as

$$\left(\begin{array}{c} \widetilde{E} \\ \widetilde{F} \end{array} \right) = \left(\begin{array}{cc} x_\xi & x_\eta \\ y_\xi & y_\eta \end{array} \right)^{-1} \left(\begin{array}{c} E(Q) \\ F(Q) \end{array} \right) = |J| \cdot \left(\begin{array}{cc} \xi_x & \xi_y \\ \eta_x & \eta_y \end{array} \right) \left(\begin{array}{c} E \\ F \end{array} \right) = \left(\begin{array}{cc} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{array} \right) \left(\begin{array}{c} E \\ F \end{array} \right) . \tag{7}$$

In the standard element, two sets of points are defined, namely the solution points (SP) and the flux points (FP). As shown in (Van den Abeele *et al.*, 2008), the stability of the method in a large array of problems is independent from the distribution of the SPs, meaning another criteria may be used in order to determine the location of these points. If, for instance, memory use is a concern, SPs could be made to coincide with FPs, hence minimizing memory allocation. In the present implementation, the favored aspects were simplicity of implementation for computational efficiency. Therefore, an internal cell discretization that only requires dealing with one-dimensional problems was selected. The use of tensor products and the enforcement that the directions of the interpolations and derivatives should coincide greatly simplify the formulation of the method. An example of such distribution, for a 3rd-order SD scheme, is illustrated in Fig. 1.

The number of points in a cell is determined by the order of the interpolating polynomial required to achieve the desired accuracy. For an n-th order method, n^2 SPs are required, such that, in each direction, there are n points and an

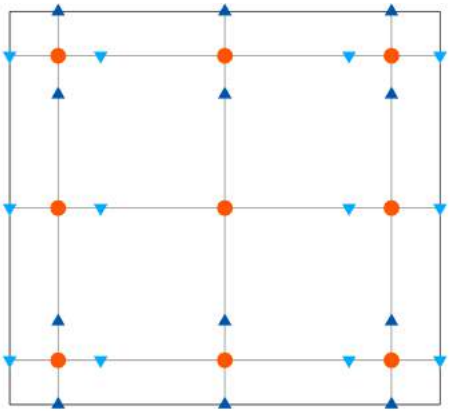


Figura 1: Possible solution point (orange circles) and flux points (blue triangles) distribution for the 3rd-order SD method.

$n-1$ degree polynomial can be reconstructed. In order to avoid the weak instability caused by using a Chebyshev-Gauss linear distribution (Van den Abeele *et al.*, 2008), the SPs are chosen to be the Gauss-Legendre points, which are defined as the roots of the Legendre polynomial of order n,

$$P_n(\xi) = \frac{1}{2^n} \sum_{s=0}^{n} \left[\binom{n}{s} (\xi - 1)^{n-s} (\xi + 1)^s \right] \tag{8}$$

shifted from the interval $[-1, 1]$ to $[0, 1]$. In order to preserve the solution accuracy, n-degree polynomials are used to interpolate the fluxes and, hence, the $n + 1$ flux points are selected to be the Legendre-Gauss-Lobatto points, defined by the roots of the Legendre polynomial of order $n - 1$ plus the end points of the interval, similarly shifted to suit the $[0, 1]$ interval.

Using the solution at n solution points, an $(n - 1)$ degree polynomial can be built using the following Lagrange basis, defined as

$$g_i(\xi) = \prod_{s=1,\, s\neq i}^{n} \left(\frac{\xi - \xi_s}{\xi_i - \xi_s} \right) . \tag{9}$$

Similarly, using the flux values at $(n + 1)$ flux points, an n degree polynomial can be built for the flux using a similar Lagrange basis, defined as

$$l_{i+\frac{1}{2}}(\xi) = \prod_{s=0,\, s\neq i}^{n} \left(\frac{\xi - \xi_{s+\frac{1}{2}}}{\xi_{i+\frac{1}{2}} - \xi_{s+\frac{1}{2}}} \right) . \tag{10}$$

The reconstructed solution for the conserved variables in the standard cell is given by the tensor product of the two 1-D polynomials,

$$Q(\xi, \eta) = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\widetilde{Q}_{i,j}}{|J_{i,j}|} \, g_i(\xi) \cdot g_j(\eta) \, . \tag{11}$$

Similarly, the reconstructed flux polynomials take the following form

$$\widetilde{E}(\xi, \eta) = \sum_{i=0}^{n} \sum_{j=1}^{n} \widetilde{E}_{i+\frac{1}{2},j} \cdot l_{i+\frac{1}{2}}(\xi) \cdot g_j(\eta), \tag{12}$$

$$\widetilde{F}(\xi, \eta) = \sum_{i=1}^{n} \sum_{j=0}^{n} \widetilde{F}_{i,j+\frac{1}{2}} \cdot g_i(\xi) \cdot l_{j+\frac{1}{2}}(\eta). \tag{13}$$

The reconstructed variables are element-wise continuous, but discontinuous across cell interfaces. In order to maintain continuity and conservation a common numerical flux must be determined for both neighboring cells. The inviscid fluxes can be directly calculated on internal FPs but require a numerical flux function in order to compute a common flux at interfaces to ensure conservation and stability. This is done using a Riemann solver to calculate a unique inviscid flux vector at cell interfaces. In this work the Roe approximate Riemann solver is used as numerical flux function for the inviscid fluxes. The numerical flux introduces artificial dissipation and, in this case, upwind characteristics to the method.

Once the polynomial interpolation for the fluxes has been constructed, the derivatives of the fluxes are computed at the solution points using the derivatives of the Lagrange operators, l, as

$$\frac{\partial \widetilde{E}}{\partial \xi} = \sum_{i=0}^{n} \sum_{j=1}^{n} \widetilde{E}_{i+\frac{1}{2},j} \cdot l'_{i+\frac{1}{2}}(\xi) \cdot g_j(\eta) \, , \tag{14}$$

$$\frac{\partial \widetilde{F}}{\partial \eta} = \sum_{i=1}^{n} \sum_{j=0}^{n} \widetilde{F}_{i,j+\frac{1}{2}} \cdot g_i(\xi) \cdot l'_{j+\frac{1}{2}}(\eta) \, . \tag{15}$$

With the inviscid fluxes uniquely defined at all FPs and interfaces, the derivatives of the sum of the fluxes are computed at the solution points using the above described derivatives of the Lagrange operators. With the gradients of the fluxes calculated on the SPs, an explicit or an implicit time-stepping method can be invoked. The explicit scheme implemented is the explicit 2rd-order, 3-stage optimal strong stability preserving Runge-Kutta scheme described in Spitieri and Ruuth (2003). Local time step is used and the characteristics length used in the CFL number calculation of each cell is considered as the mean of all four of its faces.

One important aspect of high-order methods is the representation of the boundary elements. The numerical tool developed for the present work considers a linear 2D mesh and its geometry description as inputs provided by the user. A quadratic boundary representation of such input mesh is then created for the high-order simulations. This procedure allows one to accurately represent a generic 2D geometry along with an unstructured domain discretization. Such features are necessary in order to make the high-order scheme performance comparable or superior to that of low-order methods by allowing coarser high-order meshes to be considered. Moreira *et al.* (2016) has shown the influence of high-order boundary treatment in 2D inviscid simulations as it decreases the entropy error and improves the pressure distribution along the boundaries.

3. IMPLICIT TIME-MARCHING SCHEME

3.1 Overall Implicit Scheme Formulation

The semi-discrete problem can be written for the i-th cell in the domain as

$$\frac{\partial \widetilde{Q_i}}{\partial t} = R_i \, , \tag{16}$$

where R_i is given by

$$R_i = - \left(\frac{\partial \widetilde{E}}{\partial \xi} + \frac{\partial \widetilde{F}}{\partial \eta} \right)_i \, . \tag{17}$$

The above equation can be marched in time with either an explicit or implicit time stepping algorithm. If any classical explicit time-marching scheme is considered, the Courant-Friedrichs-Lewy (CFL) number imposed by linear stability condition is often of order $O(1)$. The mesh necessary to compute viscous terms typically have highly stretched cells to account for boundary layer effects. Hence, a small time step, Δt, has to be considered to maintain numerical stability, rendering the simulation too costly for such problems. On the other hand, implicit time-marching schemes do not suffer from such stringent stability limits and they can be marched with much higher CFL number values. For instance, by applying the 1st-order accurate backward Euler method to Eq. (16), it becomes

$$\frac{Q_i^{k+1} - Q_i^k}{\Delta t} = R_i(Q^{k+1}) \, , \tag{18}$$

where k is the index of the time iteration or indicative of the time step for unsteady applications.

Equation (18) is nonlinear and, therefore, an appropriate linearization must be performed in order to allow for some efficient solver to be used. This step can be performed as

$$R_i(Q^{k+1}) \approx R_i(Q^k) + \frac{\partial R_i}{\partial Q_i} \Delta Q_i + \sum_{nb \neq i} \frac{\partial R_i}{\partial Q_{nb}} \Delta Q_{nb} \, , \tag{19}$$

resulting in the final equation

$$(\frac{I}{\Delta t} - \frac{\partial R_i}{\partial Q_i}) \Delta Q_i - \sum_{nb \neq i} \frac{\partial R_i}{\partial Q_{nb}} \Delta Q_{nb} = R_i(Q^k) \, . \tag{20}$$

Here, $\partial R_i / \partial Q_i$ and $\partial R_i / \partial Q_{nb}$ are the flux Jacobian matrices. The nb index denotes all the neighboring cells of the i-th cell and $\Delta Q = Q^{k+1} - Q^k$.

There are two main new procedures that must be performed, for an implicit scheme, in comparison to an explicit time integration method. These are the calculation of the flux Jacobian matrices and the solution of the resulting linear system. In the scope of the SD method, the flux Jacobian matrix size depends on the cell type and the order of the method. For quadrilateral cells, each Jacobian matrix is a square matrix of size $(p+1) \times (p+1) \times$ number of conserved variables, where p is the degree of the polynomial reconstruction for conserved or primitive variables. For instance, the 2-D Navier-Stokes

equations with a 2nd-order method would yield Jacobian matrices which are 16×16 matrices with the following block structure

$$\frac{\partial R_i}{\partial Q_i} = \begin{bmatrix} \frac{\partial R_1}{\partial Q_1} & \frac{\partial R_1}{\partial Q_2} & \frac{\partial R_1}{\partial Q_3} & \frac{\partial R_1}{\partial Q_4} \\ \frac{\partial R_2}{\partial Q_1} & \frac{\partial R_2}{\partial Q_2} & \frac{\partial R_2}{\partial Q_3} & \frac{\partial R_2}{\partial Q_4} \\ \frac{\partial R_3}{\partial Q_1} & \frac{\partial R_3}{\partial Q_2} & \frac{\partial R_3}{\partial Q_3} & \frac{\partial R_3}{\partial Q_4} \\ \frac{\partial R_4}{\partial Q_1} & \frac{\partial R_4}{\partial Q_2} & \frac{\partial R_4}{\partial Q_3} & \frac{\partial R_4}{\partial Q_4} \end{bmatrix} . \tag{21}$$

Here, R_j represents the residue of the j-th solution point inside the i-th cell and Q_j represents the vector of conserved variables of the j-th solution point inside the i-th cell. For the 2-D Navier-Stokes formulation, each one of these sub-matrices inside $\partial R_i/\partial Q_i$ is a 4×4 matrix. The $\partial R_i/\partial Q_{nb}$ matrix has a similar block structure.

The analytical expressions for the Jacobian matrices are quite laborious to be formulated. Therefore, the numerical calculation approach has been considered in this work, based on the following definition

$$\frac{\partial R_i}{\partial Q_i} \approx \frac{R_i(Q_{nb}, Q_i + \epsilon) \; - \; R_i(Q_{nb}, Q_i)}{\epsilon} . \tag{22}$$

Where ϵ is a small parameter, *e.g.*, $|Q_i| \times 10^{-8}$. Some caution is required in a situation where any of the conserved variables can be close zero. Such situation can occur for the momentum in the y direction in a flow essentially aligned with the longitudinal direction. In such cases, ϵ is simply made equal to 10^{-4}. This approach is the same one used by Sun *et al.* (2007a). The reader can observe that this numerical evaluation procedure is expensive, since each conserved variable of each solution point must be altered and, then, the residues of all solution points from the i-th and nb-th cells must be recalculated.

The linear system expressed in Eq. (20) is a sparse block linear system. In a 2-D quadrilateral mesh, almost every line to be solved has five non zero matrix entries, as a quadrilateral has 4 neighbors. The exceptions are the boundary cells. Therefore, the linear system has approximately $5 \times N \times [(p+1) \times (p+1) \times$ number of conserved variables$]^2$ non zero entries, where N is the total number of cells in the mesh. If chosen to be completely implicitly resolved, an appropriate linear system solver must be used such as GMRES algorithm.

3.2 GMRES Algorithm Description

The GMRES algorithm developed by Saad and Schultz (1986) is an iterative sparse linear system solver. Numerous authors (May *et al.*, 2010; Wang, 2007; Iacono *et al.*, 2010) have already used it together with high-order methods and reasonable good convergence rates have been achieved. However, this solver has important parameters that must be tuned such as the Krylov subspace size, maximum number of inner Krylov iterations allowed and the relative drop of the residue of the linear system in order to be considered as solved. All of them affects the convergence rate of the method.

Local time stepping is used, with a CFL ramp given by CFL $= 0.5 \times 2^k$, based on the work of Van den Abeele *et al.* (2009), with k being the iteration index. However, it important to emphasize that, in the beginning of the simulation, about 10 iterations have to be performed with no increase of the CFL number. Furthermore, the CFL is only increased when the linear system solver is capable of meeting the convergence criterion within the maximum number of sub-iterations allowed. Otherwise, the CFL number is actually decreased by a factor of 2. The upper limit used for the CFL number depends on the problem under consideration, GMRES algorithm parameters used and its preconditioner. Viscous computations also have a smoother CFL growth law, increasing it only every 10 iterations and by a factor of 1.1 instead of 2.0.

A very important issue related to the GMRES algorithm convergence characteristics is the preconditioner. GMRES algorithm efficiency is highly influenced by the matrix condition number (Saad, 2003) and therefore a good preconditioner should be applied to increase robustness and stability. The more the preconditioner matrix is similar to the inverse of the left-hand side matrix (LHS) defined in Eq. (20), the better is the convergence rate of GMRES algorithm. However, the preconditioner matrix should maintain the sparse pattern of the original matrix and should not have an excessive computational cost compared to the very cost of the GMRES algorithm iterations. This trade-off makes the choice of the preconditioner a difficult one to be made a priori.

The present work considers three different preconditioners: Jacobi, symmetric Gauss-Seidel (SGS) and ILU(k). The most simple of them is the Jacobi, where the preconditioner matrix is a diagonal matrix with the inverse of the elements of the original LHS matrix. SGS actually precondition the problem by making a number of symmetric sweeps with the Jacobian matrix $\partial R_i/\partial Q_{nb}$ in the right-hand side of Eq. (20). The number of sweeps is a parameter of this method. On the other hand, ILU (k) is not an iterative method. This algorithm consist of making the lower-upper factorization of the LHS matrix and inserting non-zero values only in extra k levels of fill-in. Therefore, this method tries to maintain the sparse pattern of the original matrix. In this work, the GMRES solver from the PETSc (Balay *et al.*, 2015) is used as well as the Jacobi, SGS and ILU(k) preconditioners available their.

In order to estimate the memory requirement of the GMRES algorithm, first the Jacobian matrix size must be determined. From Sec. 3,

$$memory(jacobianMatrix) = [n_{dof} \times \text{number of conserved variables}]^2 . \tag{23}$$

The variable n_{dof} corresponds to the number of degrees of freedom inside a cell and it is equal to

$$n_{dof} = (p+1) \times (p+1) \,, \tag{24}$$

where p is the degree of the polynomial reconstruction for conserved or primitive variables. Therefore, the total memory for storing $\partial R_i/\partial Q_i$ and $\partial R_i/\partial Q_{nb}$ matrices is $n_{cells} \times [n_{dof} \times \text{number of conserved variables}]^2 \times 5$ since each cell has four neighbors with the exception of boundary elements. Moreover, it is necessary to sum the extra memory for the Krylov subspace vectors. The amount of vectors n_{krylov} is one of the method parameters and each vector has $n_{cells} \times n_{dof} \times$ number of conserved variables entries. Finally, the total memory required for the GMRES algorithm is

$$\begin{aligned} memory(GMRES) = n_{cells} \times [\text{number of conserved variables}]^2 \times (p+1)^4 \times 5 + \\ n_{krylov} \times n_{cells} \times (p+1)^2 \times \text{number of conserved variables} \end{aligned} \,. \tag{25}$$

3.3 LU-SGS Method Formulation

As an effort to reduce the memory use compared to GMRES algorithm, the lower-upper factorization symmetric Gauss-Seidel (LU-SGS) method is considered. It was first applied to inviscid and viscous simulations by Sun *et al.* (2007a) and Sun *et al.* (2007b). This approach first rewrites Eq. (20) by moving the Jacobian matrix $\partial R_i/\partial Q_{nb}$ to the RHS

$$(\frac{I}{\Delta t} - \frac{\partial R_i}{\partial Q_i})\Delta Q_i^{l+1} = R_i(Q^k) + \sum_{nb \neq i} \frac{\partial R_i}{\partial Q_{nb}} \Delta Q_{nb}^* \,. \tag{26}$$

Here, the nb index denotes all the neighboring cells of the i-th cell, l is an index for the sub-iterations of the SGS algorithm, k denotes the outer iteration index, $*$ means the most recent information available (k or $k+1$) and $\Delta Q_i^{l+1} = Q_i^{l+1} - Q_i^k$.

The matrix

$$D = (\frac{I}{\Delta t} - \frac{\partial R_i}{\partial Q_i}) \tag{27}$$

has the same size as $\frac{\partial R_i}{\partial Q_i}$ in Eq. (21), $(p+1) \times (p+1) \times$ number of conserved variables. For each one of the backward and forward sweeps, the system described in Eq. (26) for the i-th cell is a dense linear system and is solved with a LU factorization. In this work, the LU algorithm in PETSc is used. Finally, in order to eliminate the need of storage of the matrices $\partial R_i/\partial Q_{nb}$, Eq. (26) can be further manipulated as follows

$$\begin{aligned} &R_i(Q^k) + \sum_{nb \neq i} \frac{\partial R_i}{\partial Q_{nb}} \Delta Q_{nb}^* = R_i(Q_i^k, Q_{nb}^k) + \sum_{nb \neq i} \frac{\partial R_i}{\partial Q_{nb}} \Delta Q_{nb}^* \approx \\ &R_i(Q_i^k, Q_{nb}^*) \approx R_i(Q_i^*, Q_{nb}^*) - \frac{\partial R_i}{\partial Q_i} \Delta Q_i^* = R_i(Q^*) - \frac{\partial R_i}{\partial Q_i} \Delta Q_i^* \,. \end{aligned} \tag{28}$$

Note that ΔQ_i^* is always the same as ΔQ_i^k for the i-th cell. Then, Eq. (26) becomes

$$(\frac{I}{\Delta t} - \frac{\partial R_i}{\partial Q_i})(\Delta Q_i^{l+1} - \Delta Q_i^l) = R_i(Q^*) - \frac{\Delta Q_i^k}{\Delta t} \,. \tag{29}$$

By seeing that $\Delta Q_i^{l+1} - \Delta Q_i^l = Q_i^{l+1} - Q_i^l$, thus

$$(\frac{I}{\Delta t} - \frac{\partial R_i}{\partial Q_i})(Q_i^{l+1} - Q_i^l) = R_i(Q^*) - \frac{\Delta Q_i^k}{\Delta t} \,. \tag{30}$$

For steady state problems, the last term in (30) can be dropped for faster convergence. Besides, the system is not marched to machine zero and actually a constant number of sweeps (3-10) is performed. Here, a sweep means one backward and one forward sweep. Lastly, it is interesting to notice in (30) that the residue in each cell must be recalculated every time the information inside i-th cell or its neighbors have changed. This approach is different than the classical Gauss-Seidel method. Local time stepping is used, with the CFL being multiplied by 1.1 every ten iterations. In the beginning of the simulation, about 10 iterations have to be performed with no increase of the CFL number. The upper limit used for the CFL number depends on the problem under consideration.

This methods only requires that the jacobian Matrix $\frac{\partial R_i}{\partial Q_i}$ is stored and there are no Krylov subspace vectors. Therefore, by using equations (23) and (24),

$$memory(LU-SGS) = n_{cells} \times [\text{number of conserved variables}]^2 \times (p+1)^4 \,. \tag{31}$$

Finally, it is possible to compare with the memory requirement of GMRES algorithm from Eq. (25)

$$\frac{memory(GMRES)}{memory(LU-SGS)} = 5 + \frac{n_{krylov}}{\text{number of conserved variables} \times (p+1)^2} \,. \tag{32}$$

In a typical inviscid 2D 2nd-order simulation with $n_{krylov} = 30$ this memory ratio is around seven. As the method‘s order increase, it decreases to the minimum value of five.

4. RESULTS

The numerical results presented here attempt to assess the performance of the high-order SD method coupled with the different techniques of convergence accelerator listed in the paper. Performance is evaluated in terms of convergence rate, CPU-time and memory use. For the results discussion, density is made dimensionless with respect to the freestream condition and pressure is made dimensionless with respect to the freestream density times the speed of sound squared. For all simulations the residue of the mass conservation equation is monitored and the tolerance considered for convergence is 10^{-12}.

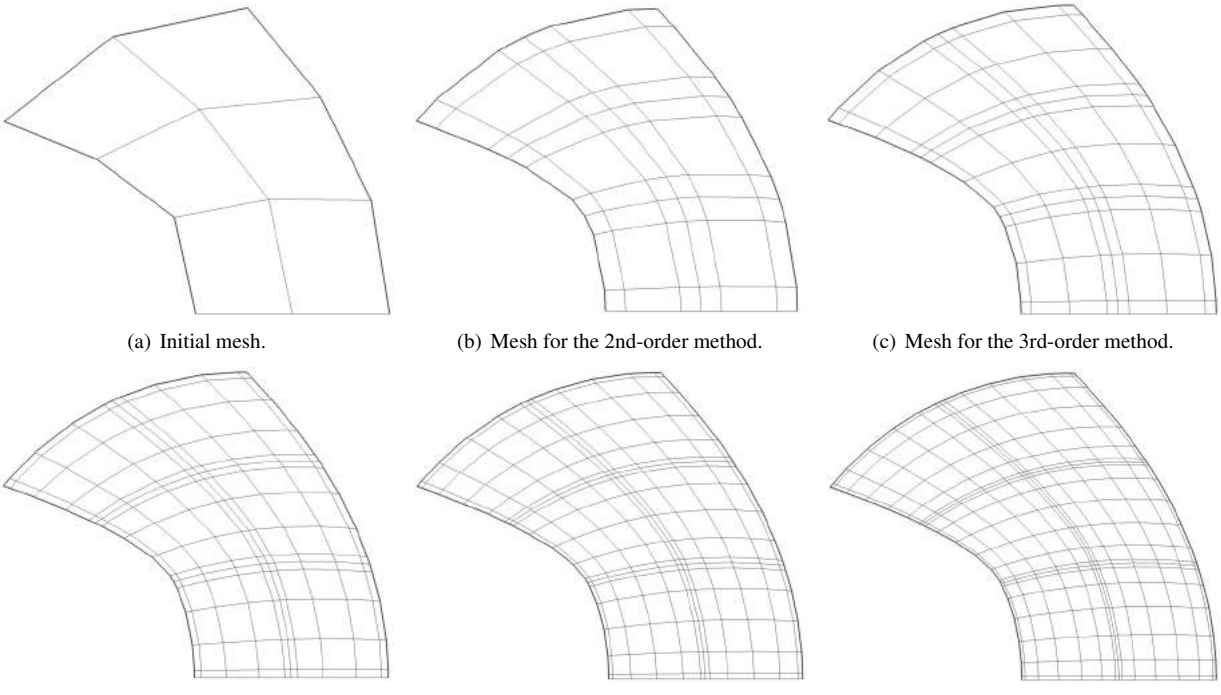
4.1 Ringleb Flow

The Ringleb flow simulation consists of an isentropic flow with an analytic solution for the Euler equations which can be derived from the hodograph transformation (Shapiro, 1953). The parameters for the case considered here are based on the test case from the International Workshop on High-Order Methods in CFD (Wang *et al.*, 2013). The flow depends on the inverse of the stream function, k, and the velocity magnitude, v_t. In the present simulations, these parameters are chosen as $k = 0.7$ and $k = 1.5$, in order to define the bounding streamlines, and $v_t = 0.5$ to define the inlet and outlet boundaries. An interesting property of the Ringleb flow test case is that transition of flow regime from supersonic to subsonic is shockless.

Implicit and explicit SD simulations, from 2nd through 6th-order, are performed to compute the solution of this problem and to assess performance in terms of convergence rate, CPU-time and memory use. Previous works (Moreira *et al.*, 2016) and (Moreira *et al.*, 2015) have already presented good results of accuracy for this same methods. The time marching schemes here considered are GMRES algorithm and LU-SGS method as discussed in Sec. 3.2 and 3.3 respectively, and also an explicit 2rd-order, 3-stage Runge-Kutta scheme.

The convergence to the analytic solution depends heavily on the accuracy of the representation of the limiting streamlines. The boundary condition of a completely tangent flow is only an accurate model of the problem if the geometric boundary sits exactly on top of an analytic streamline. The boundary condition approach adopted is to replace the tangency condition with locally computed analytic solutions, for the conserved properties, at the limiting streamlines. Therefore, even if the boundary is not in the correct location, the enforced conditions correctly represent the problem.

Figure 2 presents some of the various levels of meshes used to solve the Ringleb flow problem. Depending on the order of accuracy, each mesh element in the initial mesh, from Fig. 2(a), is further partitioned in order to account for the solution and flux points within the mesh, as in Fig. 1, to properly visualize the solution. The mesh is composed of quadrilateral cells disposed in a structured way, despite the unstructured mesh treatment by the solver. All of these meshes have a 4th-order geometric representation, but this representation can only be visualized when the flux points are added, because the visualization tool used to plot the mesh is linear.



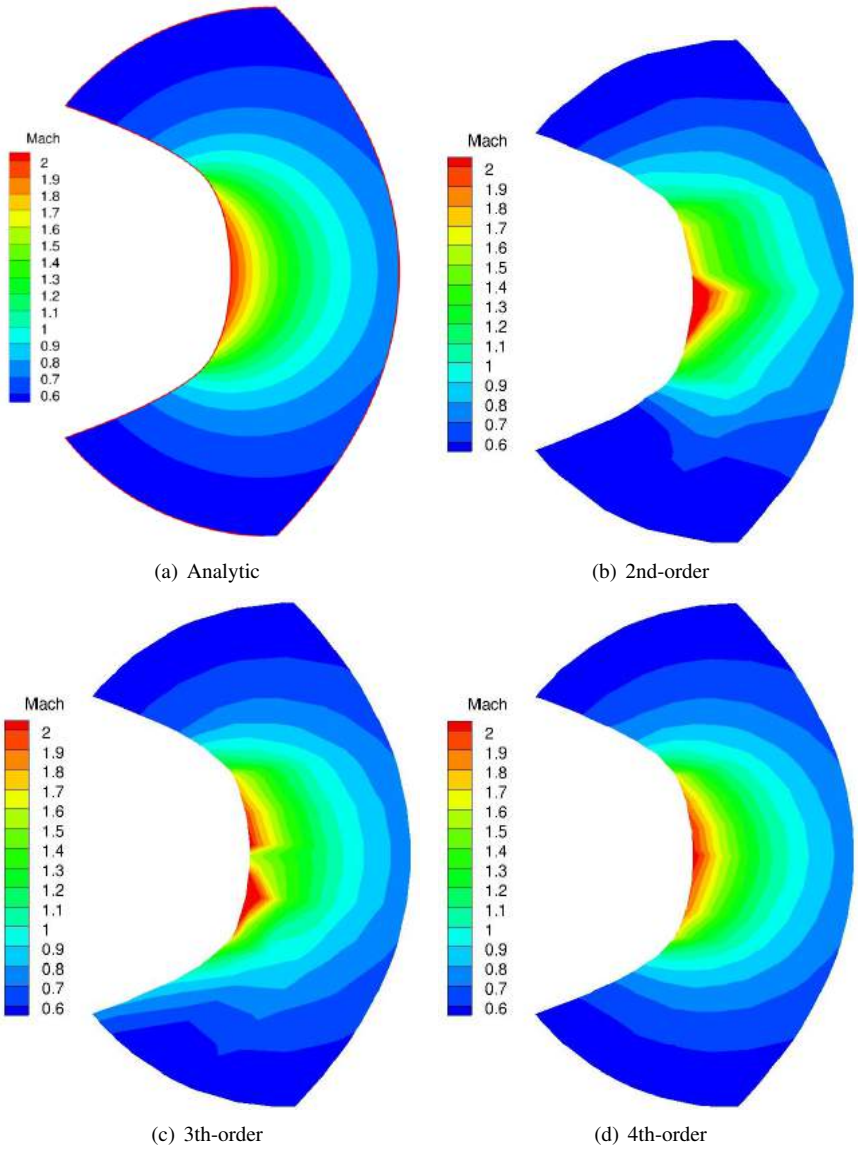
(a) Initial mesh. (b) Mesh for the 2nd-order method. (c) Mesh for the 3rd-order method.

(d) Mesh for the 4th-order method. (e) Mesh for the 5th-order method. (f) Mesh for the 6th-order method.

Figura 2: Spectral difference meshes for the Ringleb flow problem (only the upper half of each mesh is shown).

Figure 3 presents Mach number contours from 0.6 to 2.0, at equally spaced levels, for the analytic, 2nd-, 3rd- and 4th-order solutions obtained on the same meshes presented in Fig. 2. All of the present results have been computed using GMRES algorithm. It should be noticed that this is an extremely coarse grid with only 12 cells and 48, 108, 192 degrees

of freedom in the 2nd-, 3rd- and 4th-order solutions, respectively. The 2nd-order case presents severe disturbances and asymmetries, presenting a maximum entropy error of 1.36×10^{-1} and an average error of 2.04×10^{-2}. The 3rd- and 4th-order scheme results show significant improvements. The 4th-order scheme presents a maximum entropy error of 1.39×10^{-2} and an average error of order 10^{-4}. These results are essentially equal to those obtained with explicit time marching scheme discussed in Moreira *et al.* (2015).



(a) Analytic (b) 2nd-order

(c) 3th-order (d) 4th-order

Figura 3: Mach number contours for the Ringleb flow problem.

The upper limit for the CFL number in the simulations with GMRES algorithm is 10^{10}. In the cases of 2nd-order method with LU-SGS scheme the upper limit is 100 except for the last two meshes in where it is 10. With the 3rd-order SD scheme the maximum CFL used is 100 except for the last two meshes that only allowed a maximum CFL number of 5. For higher orders the maximum CFL used is also 5 except for the last mesh in where it is 2. The explicit time marching scheme achieved only a CFL number of 0.5 for the 2nd-order simulations, 0.3 for 3rd-order, 0.1 for 4th-order, 0.08 for 5th-order and 0.05 for 6th-order SD method.

For the GMRES algorithm, 30 Krylov subspace vectors is used with 250 maximum sub-iterations and a relative drop of five orders of the residue of the linear system is considered as convergence criteria. The ILU(0) is used as the preconditioner of the GMRES algorithm. The LU-SGS scheme is considered with 3 sweeps in each iteration. The results of CPU time convergence are summarized in Table 1 and it can be seen that GMRES algorithm have the best performance over the three methods. Besides, in the majority of the simulations the LU-SGS scheme outperformed the explicit scheme. It is possible that in the cases where this does not occur the maximum CFL number of the LU-SGS method could be increased without diverging.

Other important performance measure is memory use. Table 2 shows memory use for the three methods discussed: GMRES algorithm, LU-SGS method and the explicit scheme of Sec. 2. Clearly, the GMRES algorithm consumes much more memory than the others methods, reaching 8 GB in the finest mesh and highest order while LU-SGS method uses 1.7 GB and the explicit scheme 122 MB in that same case. Besides, as the SD method spatial order of accuracy increases and the mesh becomes finer, the ratio between The GMRES algorithm and LU-SGS method memory use also increases

Tabela 1: CPU time in seconds until convergence for the Ringleb flow problem using SD method with different orders of accuracy, different meshes and different time marching schemes.

Method	Cells	DOF	GMRES	LU-SGS	Explicit
2nd-order SD	48	192	0.27	0.34	1.34
	192	768	1.18	14	8
	768	3072	4.85	17	48
	3072	12288	24.57	101	310
3rd-order SD	48	432	0.97	1.27	3.41
	192	1728	4.32	83	22
	768	6912	20.37	200	165
	3072	27648	98	456	1500
4th-order SD	48	768	2.94	4.8	8.5
	192	3072	13.25	35	62
	768	12228	62	191	500
	3072	49152	310	974	3400
5th-order SD	48	1200	7.2	12	19
	192	4800	30	76	150
	768	19200	150	418	1050
	3072	76800	700	2790	5500
6th-order SD	48	1728	17	27	48
	192	6912	80	160	237
	768	27648	340	865	1550
	3072	110592	1600	4400	8000

Tabela 2: Memory use in megabytes for the Ringleb flow problem using SD method with different orders of accuracy, different meshes and different time marching schemes.

Method	Cells	DOF	GMRES	LU-SGS	Explicit
2nd-order SD	48	192	27.5	26	22.5
	192	768	33.8	30	24.1
	768	3072	59.9	47.3	29
	3072	12288	167	115	49
3rd-order SD	48	432	33.7	27.6	23
	192	1728	60.1	35.6	24.9
	768	6912	166.2	69.4	32.3
	3072	27648	591.6	203	60.8
4th-order SD	48	768	50.3	31.2	23.3
	192	3072	128.4	50	26
	768	12228	444.8	124.8	35.9
	3072	49152	1700	423.7	78
5th-order SD	48	1200	85.5	37.7	23.4
	192	4800	271.2	78.3	27.1
	768	19200	1020	236.8	41.2
	3072	76800	4022	871.5	98.3
6th-order SD	48	1728	148.5	51.9	24
	192	6912	528.5	129	28.6
	768	27648	4723	443.5	48.3
	3072	110592	8192	1690	122

reaching 4.8 with 6th-order method and the highest mesh. This value is compatible with the theoretical estimation of 5 by Eq. 32. This equation only compares the memory requirement for storing the Jacobian matrices. For low-order or coarse meshes the fraction of memory to store the other variables is significant as can be seen comparing the memory of the LU-SGS method and the explicit scheme.

4.2 NACA 0012 Subsonic Simulations

4.2.1 General Description

Inviscid subsonic NACA 0012 airfoil flow simulations are performed with for 2nd, 3rd and 4th-order SD method using the GMRES algorithm, LU-SGS method and an explicit scheme with a linear mesh. The freestream Mach number value

of $M_\infty = 0.6$ and $\alpha = 0$ deg. angle-of-attack are used, which ensure a smooth flow throughout the domain. The original Q1 mesh is composed of 4619 quadrilateral cells and is presented in Fig. 4. The geometry of the airfoil is modified to have a collapsed trailing edge. The farfield boundary is located at a distance of 1000 chords from the airfoil, using a *C*-type grid topology. A quadratic Q2 mesh was obtained from the original Q1 mesh.

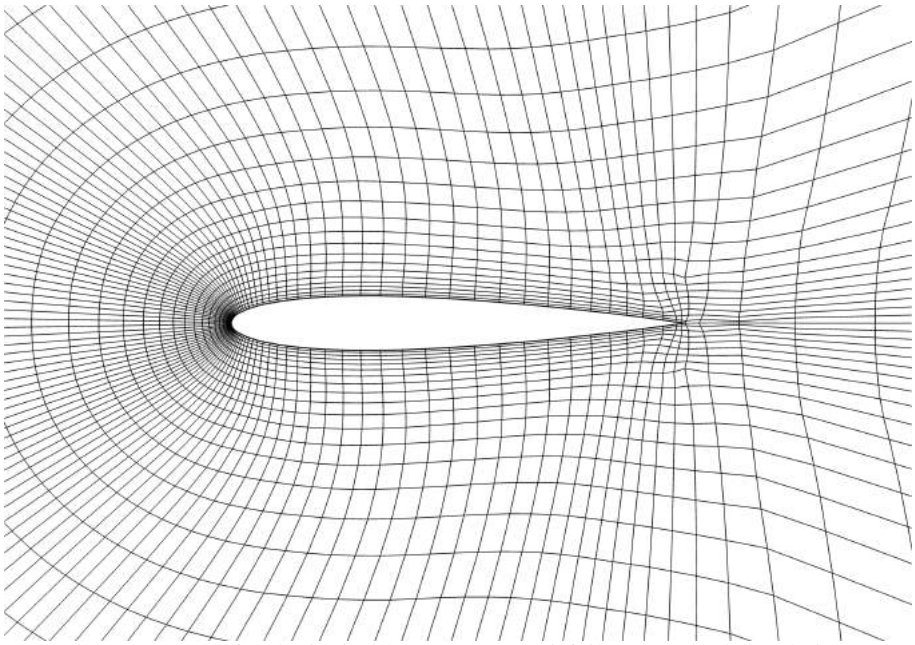
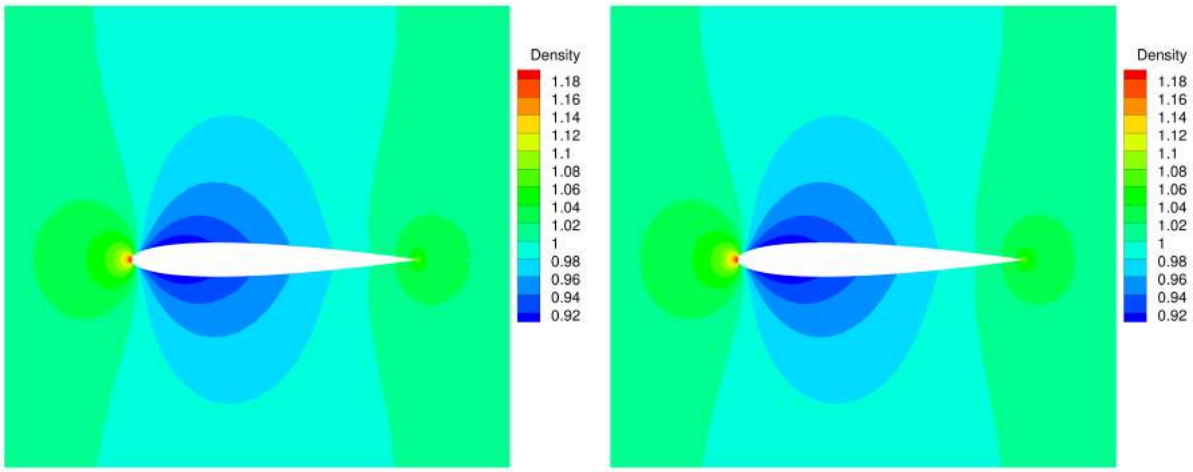


Figura 4: Mesh for the inviscid NACA 0012 airfoil SD method simulation.

Figure 5 shows the density contours for the NACA 0012 subsonic simulations. The flow property contours share the same color map range and number of levels. Hence, a qualitative comparison can be made from the solutions at different orders of accuracy. There are some very small differences between the 3rd-order solution and the 2nd-order one. In particular, the 3rd-order method solution seems to have slightly smoother contour lines than the results obtained with the 2nd-order method. Previous work (Moreira *et al.*, 2016) have showed the effect of using a high-order representation for the mesh, decreasing entropy error at airfoil surface and improving the pressure distribution inside the cells near the airfoil walls.



(a) 2nd-order. (b) 3rd-order.

Figura 5: Density contours for the NACA 0012 airfoil inviscid simulations at $M_\infty = 0.6$ and $\alpha = 0$ deg.

4.2.2 Comparison Between GMRES and LU-SGS Methods

Simulations are performed with 2nd, 3rd and 4th-order SD methods with the GMRES algorithm, the LU-SGS method and an explicit scheme. The mesh used is the one described in Fig. 4. The GMRES algorithm parameters are: ILU(0) as preconditioner, 100 for the size of the Krylov subspace, 250 maximum sub-iterations and five orders of magnitude as the relative convergence criteria for the linear system solution. For the LU-SGS scheme three sweeps are performed in each iteration. The CFL upper limit is 10^{10} for the GMRES algorithm, 2 for the LU-SGS method and 0.05 for the explicit scheme.

The results of CPU time and iterations to converge are summarized in Table 3 and it is possible to see that the GMRES algorithm have the best performance in terms of time to converge and the explicit scheme have the worst one. Both GMRES algorithm and LU-SGS method converge in a number of iterations almost independent of the SD method order of accuracy, around 30 and 1660 respectively. The explicit scheme does not show this characteristic. On the other hand, Table 4 shows the results for memory use and again as well as in Sec. 4. lthe GMRES algorithm have the worst performance, using 2.5 GB in the 4th-order method simulation. It is possible to verify that for the 3rd-order method the ratio between GMRES algorithm and LU-SGS scheme memory use is 3.0 and for the 4th-order method it is 4.13, results that are compatible with Eq. 32. As the method‘s order increases, the fraction of the total memory used to store the Jacobian matrix also increases.

Tabela 3: CPU time and iterations to converge the NACA 0012 inviscid subsonic simulations for different orders of accuracy and time marching schemes.

Method	Cells	DOF	Iterations			CPU time (s)		
			GMRES	LU-SGS	Explicit	GMRES	LU-SGS	Explicit
2nd-order SD	4619	18476	30	1653	120608	39	1332	6638
3rd-order SD	4619	41571	31	1661	168043	200	3872	19400
4th-order SD	4619	73904	31	1663	230550	617	10300	45720

Tabela 4: Memory use in megabytes of the NACA 0012 inviscid subsonic simulations with different orders of accuracy and time marching schemes.

Method	Cells	DOF	GMRES	LU-SGS	Explicit
2nd-order SD	4619	18476	225	148	52
3rd-order SD	4619	41571	865	283	69
4th-order SD	4619	73904	2535	613	92

4.2.3 Evaluation of GMRES and LU-SGS Method Parameters

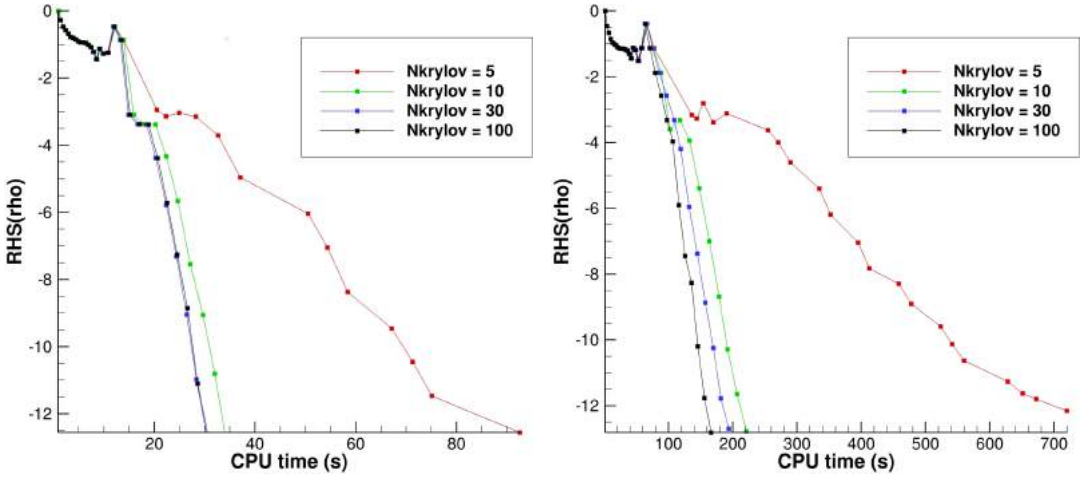
Simulations are performed with 2nd, 3rd and 4th-order SD methods with GMRES algorithm and varying the amount of Krylov subspace vectors. The comparison of CPU-time for the results are shown in Fig. 6. The maximum number allowed of sub-iterations for the GMRES algorithm is 250 and the linear system is considered converged after a relative drop of 5 orders in magnitude of the residue. The upper limit for the CFL is 10^{10} and the ILU(0) is used as the preconditioner. The simulations with a Krylov subspace of size 5 and 10 have poor convergence characteristics for all orders considered. In those cases the sub-iterations number achieved the maximum of 250 without convergence and therefore the CFL increases slower. On the other hand, both Krylov subspace of size 30 and 100 have similar total CPU-time for convergence of all orders, with just a minor increase in performance in the 3rd-order case by using 100 Krylov subspaces.

The same NACA 0012 subsonic case is simulated with 2nd, 3rd and 4th-order SD method and GMRES algorithm but now varying the amount of orders the linear system residue must drop to be considered converged. The comparison of CPU-time for the results are shown in Fig. 7. The maximum number allowed of sub-iterations for the GMRES algorithm is 250 and a size of 30 is used for the Krylov subspace. The upper limit for the CFL is 10^10 and again the ILU(0) is used as the preconditioner for the GMRES algorithm. The cases simulated have a relative drop of one, three, five and seven orders of magnitude. The cases that are not shown have diverged, for example, the case with a relative drop of three orders of magnitude with the 3rd-order SD scheme. Only the test cases with a relative drop of five and seven orders of magnitude have converged for all SD orders, with better performance for the relative drop of five orders.

Lastly, the only parameter of the LU-SGS method is the number of sweeps in each iteration. It is varied from 1 to 10. In this case only the 3rd-order SD method is simulated and the results of CPU time and number of iterations are shown in Fig. 8. It can be seen in Fig. 8(a) that increasing the number of sweeps also increases the convergence rate. The linear system of each iteration is better resolved and therefore less iterations are needed. On the other hand, it is visible in Fig. 8(b) that for the simulations with the same CFL number, three sweeps performed better in terms of CPU time. Making ten sweeps per iteration does not pay off the increase of time per iteration. Furthermore, almost all cases have an upper limit for the CFL number of 2. However, by making ten sweeps each iteration the CFL number can be increased to 20 and still converges. The performance in this case also gets 10% better in CPU time than the one with three sweeps. Nevertheless, the maximum upper limit for CFL that still converges is a case by case try and error task that demands time and making three sweeps performed almost as better than all the other simulations.

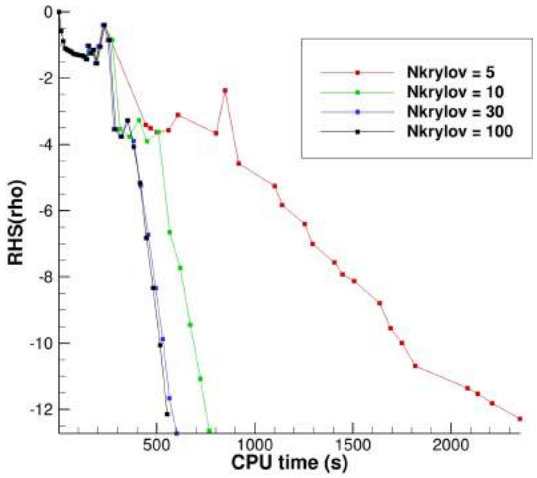
4.2.4 Preconditioner Evaluation

Simulations are performed with 2nd, 3rd and 4th-order SD methods with GMRES algorithm and varying the preconditioner. All simulations used ILU(k), but varying the level k of fill in: zero, one, two and four. The comparison of



(a) 2nd-order.

(b) 3rd-order.



(c) 4th-order.

Figura 6: Convergence history in CPU-time of the the NACA 0012 airfoil inviscid simulations varying the amount of Krylov subspace vectors.

CPU-time are shown in Fig. 9. The maximum number allowed of sub-iterations for the GMRES algorithm is 250 and the linear system is considered converged after a relative drop of 5 orders in magnitude of the residue. The upper limit for the CFL is 10^{10} and the amount of Krylov subspace vectors is 30.

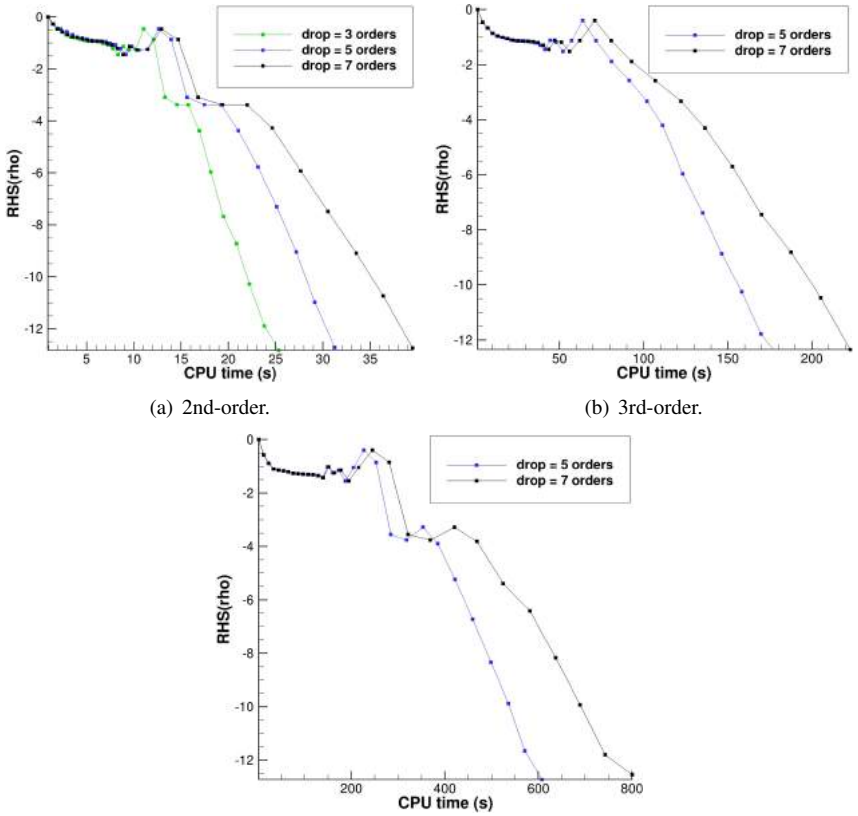
For both 2nd- and 3rd-order SD methods, ILU(1) outperformed the others preconditioners. ILU(2) and ILU(4) are more similar to the ideal preconditioner matrix but the extra computational cost per iteration do not pay off in convergence rate. As the methods order increases, ILU(0) becomes more competitive with ILU(1) and finally in the 4th-order case the ILU(0) is the fastest to converge. In Table 5 it is possible to see that the preconditioner also influence the memory use. In the 2nd-order method the difference from ILU(0) and ILU(4) is 70 MB and in the 4th-order case it reaches 1.1 GB.

Tabela 5: Memory use in megabytes of the NACA 0012 inviscid subsonic simulations with different orders of accuracy and different preconditioners.

Method	Cells	DOF	ILU(0)	ILU(1)	ILU(2)	ILU(4)
2nd-order SD	4619	18476	225	241.5	259	295
3rd-order SD	4619	41571	865	956	1043	1218
4th-order SD	4619	73904	2535	2824	3099	3650

5. CONCLUDING REMARKS

The present work discusses several issues and results for an implementation of the high-order SD method for inviscid compressible flows. The SD method uses a simple universal reconstruction to yield high-order polynomial approximations of the solution. Accuracy evaluation, such as that using the Ringleb flow problem, demonstrates the capability of the method to provide solutions with the expected accuracy. The main contribution of the present work is to evaluate the performance of different convergence accelerators coupled to implicit time integration schemes, such as the GMRES and LU-SGS methods and when applied to high-order methods. The performance is measured in terms of convergence rate,

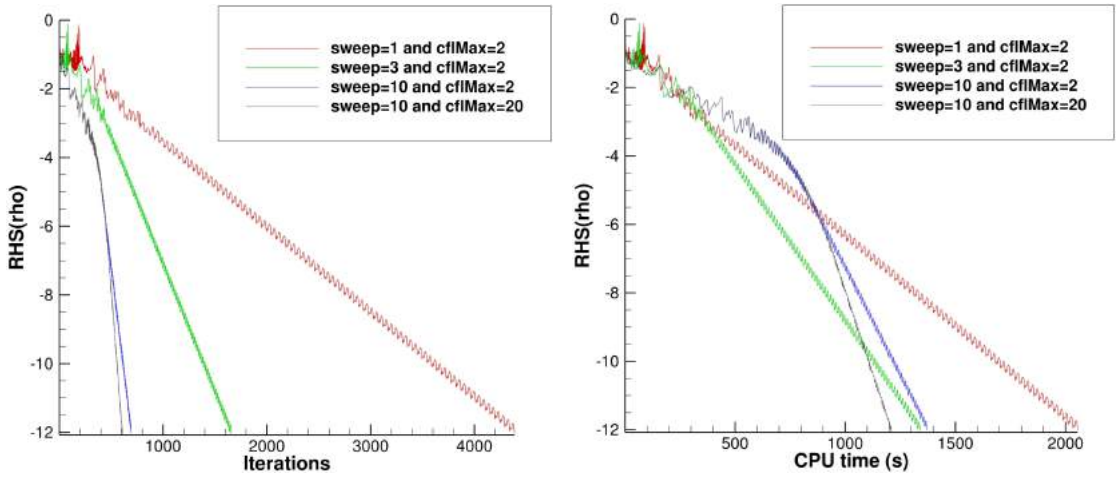


(a) 2nd-order.

(b) 3rd-order.

(c) 4th-order.

Figura 7: Convergence history in CPU-time of the the NACA 0012 airfoil inviscid simulations varying the amount of orders the linear system residue must drop to be considered converged.



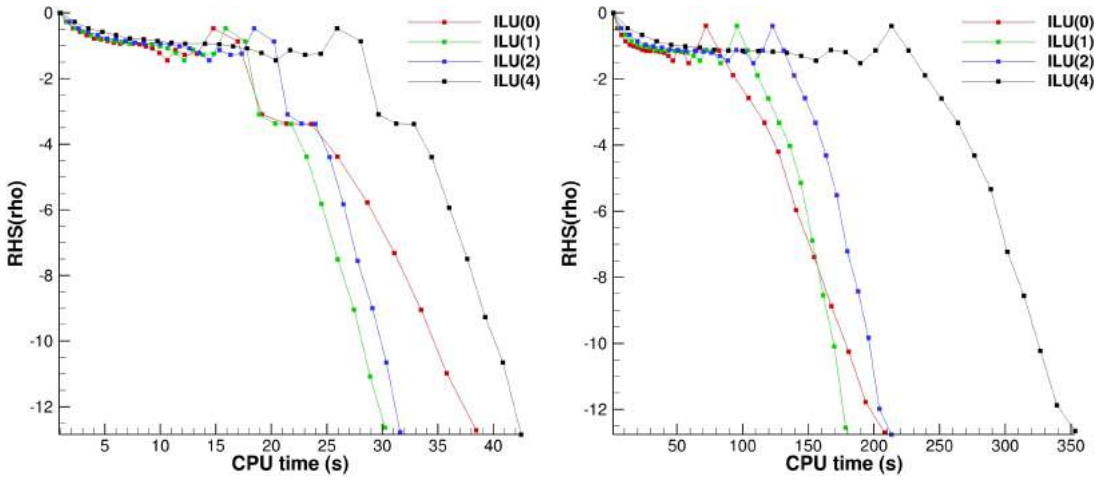
(a) Number of iterations.

(b) CPU time.

Figura 8: Convergence history of the the NACA 0012 airfoil inviscid simulations varying the number of sweeps inside each iteration of the LU-SGS method for the 3rd-order SD method.

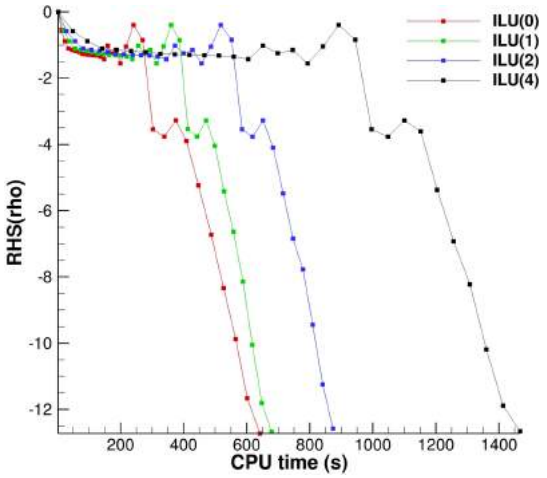
CPU time and memory usage.

The SD method implemented delivers the expected results for subsonic inviscid and flows. The GMRES algorithm seems to be the best method tested in terms of CPU time to convergence of 2-D SD methods from 2nd- to 6th-order accuracy. On the other hand, it uses a large amount of memory as the method order increases and, if there is any memory constraint in the machine, then the LU-SGS scheme should be used. In the cases tested so far, it offers the best compromise regarding convergence rate and memory use. For inviscid simulations, the GMRES algorithm parameters, that have shown the best performance so far, consider a Krylov subspace size of 30 and five orders of magnitude of relative residue drop for the linear system to be considered converged. When using local time stepping, the CFL number ramp is found to be very important and it considerably influences the performance of the methods, especially for the LU-SGS scheme. It is clear that efforts directed towards convergence rate improvements and robustness are fundamental for the efficient use of high-order methods. The authors intend to continue exploring those issues and to carry out related techniques in more complex, three dimensional problems in the future.



(a) 2nd-order.

(b) 3rd-order.



(c) 4th-order.

Figura 9: Convergence history in CPU-time of the the NACA 0012 airfoil inviscid simulations varying the preconditioner.

6. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support for the present research provided by Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, under the Research Grants No. 309985/2013-7, No. 400844/2014-1 and No. 443839/2014-0. The work is also supported by Fundação de Amparo à Pesquisa do Estado de São Paulo, FAPESP, under Research Grant No. 2013/07375-0. The support provided by Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, CAPES, through graduate scholarships for all graduate student authors, is also greatly appreciated.

7. REFERENCES

Balay, S., Abhyankar, S., Adams, M.F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, W.D., Kaushik, D., Knepley, M.G., McInnes, L.C., Rupp, K., Smith, B.F., Zampini, S. and Zhang, H., 2015. "PETSc Web Page". `http://www.mcs.anl.gov/petsc`. URL `http://www.mcs.anl.gov/petsc`.

Iacono, F., May, G. and Wang, Z.J., 2010. "Relaxation techniques for high-order discretizations of steady compressible inviscid flows". In AIAA Paper No. 2010-4991*, 40th Fluid Dynamics Conference and Exhibit*. Chicago, IL.

May, G., Iacono, F. and Jameson, A., 2010. "A hybrid multilevel method for high-order discretization of the Euler equations on unstructured meshes". *Journal of Computational Physics*, Vol. 229, No. 10, pp. 3938–3956.

May, G. and Jameson, A., 2006. "A spectral difference method for the euler and navier-stokes equations on unstructured meshes". In AIAA Paper No. 2006-0304, *Proceedings of the 44th AIAA Aerospace Sciences Meeting*. Reno, NV.

Moreira, F.M., Breviglieri, C. and Azevedo, J.L.F., 2015. "High-order compressible flows simulations with the unstructured spectral difference method". In AIAA Paper No. 2015-3195*, Proceedings of the 22nd AIAA Computational Fluid Dynamics Conference*. Dallas, TX.

Moreira, F.M., Jourdan, E., Breviglieri, C., Aguiar, A.R.B. and Azevedo, J.L.F., 2016. "Implicit spectral difference method solutions for compressible flows considering high-order meshes". In *To appear in the Proceedings of the 23nd AIAA Computational Fluid Dynamics Conference*. Washington, D.C.

Saad, Y., 2003. *Iterative Methods for Sparse Linear Systems*. SIAM.

Saad, Y. and Schultz, M.H., 1986. "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear

systems". *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, pp. 856–869.

Shapiro, A.H., 1953. *The Dynamics and Thermodynamics of Compressible Fluid Flow*. Wiley, New York.

Spitieri, R.J. and Ruuth, S.J., 2003. "A new class of optimal high-order strong-stability-preserving time discretization methods". *SIAM Journal on Numerical Analysis*, Vol. 40, No. 2, pp. 469–491.

Sun, Y., Wang, Z.J. and Liu, Y., 2007a. "Efficient implicit non-linear LU-SGS approach for viscous flow computation using high-order spectral difference method". In AIAA Paper No. 2007-4322*, Proceedings of the 18th AIAA Computational Fluid Dynamics Conference*. Miami, FL.

Sun, Y., Wang, Z.J., Liu, Y. and Chen, C.L., 2007b. "Efficient implicit non-linear LU-SGS algorithm for high-order spectral difference method on unstructured hexahedral grids". In AIAA Paper No. 2007-0313*, Proceedings of the 45th AIAA Aerospace Meeting*. Reno, NV.

Van den Abeele, K., Lacor, C. and Wang, Z.J., 2008. "On the stability and accuracy of the spectral difference method". *Journal of Scientific Computing*, Vol. 37, No. 2, pp. 162–188.

Van den Abeele, K., Parsani, M. and Lacor, C., 2009. "An implicit spectral difference navier-stokes solver for unstructured hexahedral grids". In AIAA Paper No. 2009-0181*, Proceedings of the 47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*. Orlando, FL.

Wang, Z.J., 2007. "High-order methods for the euler and navier-stokes equations on unstructured grids". *Progress in Aerospace Sciences*, Vol. 43, pp. 1–41.

Wang, Z.J., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H.T., Kroll, N., May, G., Persson, P., van Leer, B. and Visbal, M., 2013. "High-order cfd methods: Current status and perspective". *International Journal for Numerical Methods in Fluids*, Vol. 72, No. 8, pp. 811–845.

Wang, Z.J., Liu, Y., May, G. and Jameson, A., 2007. "Spectral difference method for unstructured grids ii: Extension to the euler equations". *Journal of Scientific Computing*, Vol. 32, No. 1, pp. 3938–3956.

8. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.