CERFACS

ISAE Institut Supérieur de l'Aéronautique et de l'Espace ENSMA

Unsteady simulations with a high-order CFD code

Author : David De Oliveira Amorim

Supervisors : G. Puigt, J.-F. Boussuge
ISAE-ENSMA Supervisor : L. Perault

Ref.: WN/CFD/15/06
January 6, 2015

Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique
42 avenue Coriolis – 31057 Toulouse Cedex 1 – France
Tél : +33 5 61 19 31 31 – Fax : +33 5 61 19 30 00
http://www.cerfacs.fr – e-mail: secretar@cerfacs.fr

Contents

Chapter 1

Introduction

1.1 General Considerations

During the last 40 years, the importance of Computational Fluid Dynamics (CFD) has grown, first in university labs and then in industry. In labs, the development followed the beginning of computers and then supercomputers and the emergence of dedicated schemes to account for convection or diffusion effects. Industry found immediately the interest of CFD: with CFD, equations are discretized and solved on a given mesh and CFD induces the possibility to handle easily several shapes, something impossible in a short time with experiences. Parametric studies are possible with CFD. Moreover, CFD offers the possibility to have information at all time instants and everywhere inside the domain of interest, which is impossible during expemiments. For all those reasons, industries have made many efforts to introduce, to validate and to define best practices for their configurations.

Industrial flows are generally transonic and turbulent. Turbulence is one of the most complex and mathematically badly-understood phenomena. Turbulence is characterized by vortices of different size and energy that are produced, convected and dissipated by the flow. The largest sizes are linked with the object size, while the lowest sizes are dissipated by viscous effects.

At the beginning of CFD, some flows could not be solved without modeling: the computational capabilities were too limited. Following the emergence of Reynolds Averaged Navier-Stokes (RANS) equations, many efforts were dedicated to solve this approach and nowadays, industries are able to perform RANS or unsteady RANS computations every day. The basic schemes are second-order accurate and dissipative for stability and robustness. However, the averaging procedure hides all unsteady turbulence effects and, now, the tendency is to account for the unsteadiness of turbulent effects, which is currently one of the main objectives at CERFACS.

1.2 CERFACS

CERFACS (Centre Européen de Recherche et de Formation avancée en Calcul Scientifique) is a research organization that aims to develop advanced methods for the numerical simulation and the algorithmic solution of large scientific and technological problems of interest for research as well as industry, and that requires access to the most powerful computers presently available. CERFACS is governed by a Conseil de Gérance with representatives from its shareholders, and benefits from the recommendations of its Scientific Council.

CERFACS has seven shareholders : CNES, the French Space Agency; EADS France, European Aeronautic and Defence Space Company; EDF, Electricité de France; Météo-France, the French meteorological service; ONERA, the French Aerospace Lab; SAFRAN, an international high-technology group; TOTAL, a multinational energy company.

CERFACS hosts interdisciplinary teams, both for research and advanced training that are composed of: physicists, applied mathematicians, numerical analysts, and software engineers.

Approximately 150 people work at CERFACS, including more than 130 researchers and engineers, coming from 10 different countries. They work on specific projects in eight main research areas: parallel algorithms, code coupling, aerodynamics, gas turbines, combustion, climate, environmental impact, data assimilation.

1.3 CFD Team

The CFD Team represents approximately half of the CERFACS, both in human resources and in financial support. The CFD team is organised with three main components: combustion, turbo machinery and applied aerodynamics. The objective of the CFD group at CERFACS is to solve problems involving both CFD and High Performance Computing (HPC). Despite the recent progresses observed in CFD, the solution of many flows of interest is still beyond present capabilities and the challenge of HPC for CFD remains as open and difficult as it ever was. In most CFD problems, brute force approaches still fail and advances in this field rely on defining proper compromises between physics and numerics.

This is especially true in the fields of CFD chosen at CERFACS: aerodynamics, turbulence, combustion, unsteady flows, coupled phenomena between fluid mechanics and other mechanisms (fluid structure interaction, optimisation, two-phase flows, radiation, etc).

In the last years, the requests of CERFACS partners as well as the general orientation of the CFD community have lead the CFD project to a deeper implication in Direct Numerical Simulation tools, especially for reacting flows or for flows in complex geometries as well as to the development of new aspects of CFD such as multiphysics or active control. This has been done through an increase of the CFD staff so that the classical expertise of the CFD team (aerodynamics, turbulence modelling, optimisation and parallelisation, combustion) has been maintained or even reinforced. An important new field of application for CERFACS is Large Eddy Simulation (LES). The role of the CFD team and of its partners in the development of Large Eddy Simulation is now significant through multiple collaborations, contracts and dissemination of information and tools. The LES approach has emerged as a prospective technique for problems associated with time dependent phenomena and coherent eddy structures. This leading edge CFD technology can nowadays be applied to geometries of reasonable complexity (such as a combustion chambers in gas turbines but also in piston engines), which is the result of both constantly increasing computer capacities along with improved underlying numerical methods and grid techniques. In 2014, more than 60 scientists were working in the CFD project at CERFACS.

1.4 A new discretisation approach for accurate CFD simulations: the Spectral Difference Method

Developing accurate schemes (for LES) depends highly on the mesh chosen for the computations. Even if classical approaches (implicit compact, Finite Difference, WENO... schemes) are easy to implement in a structured framework, the situation is much more complex for unstructured grids. In particular, any approach with large stencil must be avoided for several reasons. First, computing large stencil needs complex geometric algorithms to implement and to validate and specific situations can be encountered, which may lead to implement correction(s). In particular, such a situation generally occurs when the mesh is composed of elements with different shapes. Moreover, large stencil and High Performance Computing (HPC) are two topics with opposite objectives. It is clear that many fields must be exchanged when a large stencil is considered and therefore messages have large length. The sequential efficiency is also impacted and if data are badly arranged in memory, cache misses lead to a high CPU time per degree of freedom. Finally, the efficiency of such codes is rather connected with memory bandwidth and MPI efficiency (of the supercomputer) than with the kernel itself. But unstructured grids are unavoidable for computations over complex CAD and a lot of work has been published during the last years on high order schemes for unstructured grids.

A way to overcome large stencils on unstructured grids lies on increasing the number of degrees of freedom inside the element. It is convenient to define high order representations of quantities inside any mesh element following a polynomial approximation. The reconstructed variables at the faces depend on the mesh element only and two different extrapolations (one at left side and one at right one) may lead to discontinuous flow at mesh faces. Among the methods proposed in the literature, we have identified three techniques:

- The Discontinuous Galerkin -DG- technique is based on the Finite Element framework. The principle is to look for a polynomial representation of the solution that satisfies a variational form of the governing system within each element. Even if the technique is quite old (Reed and Hill in 1973 [12]), its extension to the full Navier-Stokes equations is recent and many papers have been published during the last 10 years.

- The Spectral Volume -SV- technique is based on the Finite Volume framework and it follows the pioneering work of Wang in 2002 [21, 22]. It consists in defining element subdivisions on which a classical Finite Volume technique is considered. The mean quantity over each volume is necessary to build the high order representation of data inside the element.

- The Spectral Difference -SD- technique follows the Finite Difference approach. Kopriva and Kolias published it in 1996 [4] and Liu, Vinokur and Wang [7] published a more general presentation of the technique in 2006. The idea is to define high order approximation of the quantities but to solve the strong form of the equations, as in Finite Differences, inside each mesh cell.

All techniques define high-order continuous solution inside each mesh element and since the reconstruction leads to two different quantities at mesh interface, a Riemann solver is necessary to compute the flux to exchange between cells. The interest of all methods comes also from the possibility to manage both the space refinement parameter h and the degree of the polynomial p. When one compares classical Finite Volume technique with those high-order

ones, the main difference lies in the non-universal relation between the mesh element and the number of degrees of freedom (1 mesh element is not associated with 1 degree of freedom as in Finite Volume). This point has a (very) strong impact on the high order unstructured data structure.

The Spectral Difference technique has been implemented 2 years ago in a new CFD code called JAGUAR (proJect of an Aerodynamic solver using General Unstructured grids And high ordeR schemes). The choice of the SD approach is motivated by several reasons: the SD method has been built in order to correct some drawbacks of DG and SV. First, it seems more efficient in term of CPU usage (less computations per degree of freedom) than DG technique. Moreover, SV suffers a high sensitivity with respect to element decomposition and this drawback is avoided with SD method. Finally, the SD approach is less mature than DG and the potential of research work is greater.

1.5 Description of the work and outline of the document

This report follows a three-month internship at CERFACS in the CFD Team. There was initially two goals. The first objective concerned the validation of several enhancements implemented in JAGUAR in order to increase its accuracy or to decrease the CPU time for the simulations. The second goal is to run, post-treat and analyse a LES of a plane channel.

This report is composed of two main parts. First, SD method principles will be introduced, followed by the presentation of the results concerning the 2D convected vortex simulations.

Chapter 2

Spectral Difference method

In this chapter, the SD approach is introduced in 1D for a hyperbolic equation and explanations are given to justify 2D and 3D flow extension. Moreover, even if this report essentially addresses hyperbolic equations leading to discontinuity, the SD diffusion scheme is also introduced as it will be considered for LES simulations.

2.1 Notations and General principle

The following basic hyperbolic equation is considered to introduce the principle of the Spectral Difference (SD) method in 1D:

$$\frac{\partial Q}{\partial t} + \frac{\partial \mathcal{F}}{\partial x} = 0, \tag{2.1}$$

and a mesh composed of several 1D segment.

The principle of the SD method is to assume that the vector of unknowns Q varies as a polynomial with a predefined degree p inside each segment of the mesh. To define the p-order polynomial, the vector Q must be of size $p+1$ and $p+1$ points, which are called solution points, are defined inside each segment. The solution is known in solution points and the numerical values at solution points help to define the polynomial approximation representation of unknown. As for the Finite Difference method, solution is computed in any i degree of freedom, corresponding here to the solution points:

$$\frac{\partial Q_i}{\partial t} + \left.\frac{\partial \mathcal{F}}{\partial x}\right|_i = 0.$$

Following an explicit time marching process, the solution evolution is known once the derivative of the flux $\mathcal{F}$ is computed at each solution point. Let consider an example. Take $p = 2$ in any segment of the mesh. The solution points, the unknowns in solution point and the second-order interpolation polynomial are introduced in Fig. 2.1. The polynomial representation is by essence local: it is associated with a given segment. By nature, the SD method does not assume that polynomials are continuous at the interface between two segments.

Following Eq. 2.1, it is mandatory that the gradient of flux $\mathcal{F}$ is a $p-th$ order order polynomial and the interpolation polynomial of the flux $\mathcal{F}$ must be $(p+1)-th$ order accurate. The flux polynomial must be defined on $p+2$ points that are called flux points. The classical SD method is staggered and the "external" flux points are located at the two end points of

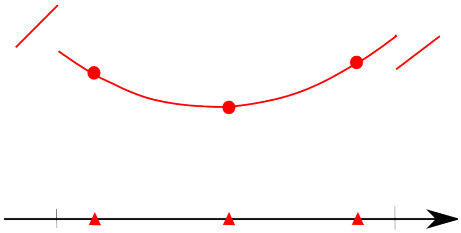

Figure 2.1: Step 0: Solution points (▲), solution (●) and 2^{nd} order polynomial on the segment

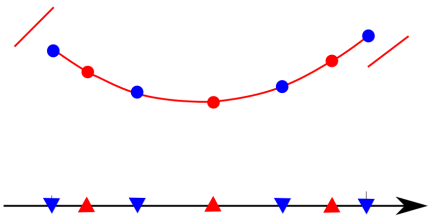

Figure 2.2: Step 1: Extrapolation of solution (●) at the flux points (▼)

any segment and other flux points are introduced between two contiguous solution points. For $p = 2$, 4 flux points are illustrated in FIG. 2.2. The flux is a function of the state at the flux point and the interpolation polynomial from solution points is considered to define the solution in the flux points, as in FIG. 2.2.

The flux density $\mathcal{F}$ is obtained as a non linear combination of the values Q at the internal flux points, FIG. 2.3. At the interface, the flux points are shared by two segments and since values are discontinuous at interfaces, the flux is not defined without ambiguity. But the flux can be estimated by the solution of a Riemann problem using the interface value as inputs and the flux as output. At the present time, the flux is then defined with uniqueness in all flux points, FIG. 2.4.

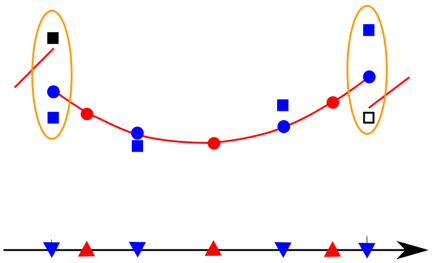

Figure 2.3: Step 2: Computing flux $\mathcal{F}$ (■) at flux points

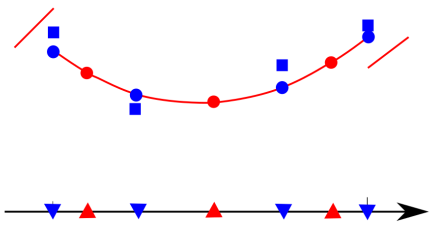

Figure 2.4: Step 3: unique flux computing in segment at interface with a Riemann solver

From flux density, one can build a $(p+1) - th$ degree interpolation polynomial, FIG. 2.5. The polynomial is globally continuous but only differentiable inside each segment and not at the interfaces.

The last step consists in differentiating the polynomial to compute the increment at each solution point, FIG. 2.6.

2.1.1 Position of the solutions points and the flux points

The solution points are chosen to be the following Gauss points:

$$X_s = \frac{1}{2}\left[1 - cos\left(\frac{2s-1}{2N}\pi\right)\right] \text{ with } s = 1, \ldots, N. \tag{2.2}$$

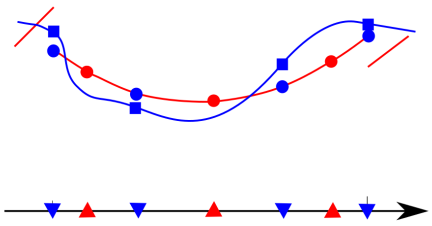

Figure 2.5: Step 4: Building a new polynomial at flux points (at degree $p+1$)

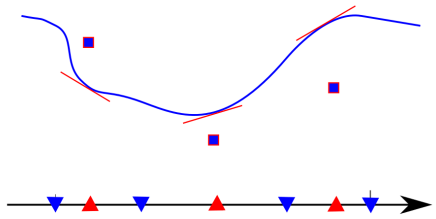

Figure 2.6: Step 5: Differentiation of the flux polynomial at solution points

In the following, flux points will be defined with a non integer index, a flux point numbered $s+1/2$ is located between solution points s and $s+1$. The flux points are defined as the roots of Legendre polynomial, a serial of polynomial of increasing order defined by the following ordinary differential equation on $[-1,1]$:

$$\frac{d}{dx}\left((1-x^2)P_n'(x)\right) + n(n+1)P_n(x) = 0, \quad P_n(1) = 1. \tag{2.3}$$

The solution points and the flux points positions are described in FIG. 2.7 for $p=2$.

Figure 2.7: Position of solution points (▼) and flux point (▲) for $p=2$

In the literature, there is another choice to determine the flux points, they are defined as the Gauss-Lobatto points:

$$X_{s+1/2} = \frac{1}{2}\left[1 - cos\left(\frac{s}{N}\pi\right)\right] \text{ with } s = 0, 1, \ldots, N. \tag{2.4}$$

A. Jameson [3] studied the proof of the Spectral Difference method stability. The solution points location does not affect the stability nor the accuracy of the method for a linear hyperbolic equation. But, the study of K. Van Den Abeele [20] shows that flux points location has a strong influence on both stability and accuracy of the method. He proves that the use of Gauss-Lobatto points is not a good choice for the flux points, the resulting SD schemes being unstable for orders of accuracy higher than 2. The Legendre flux points are considered for this work.

2.1.2 Polynomials

Using the solutions at N solutions points, where $N = p+1$, a p-degree polynomial can be built using the following Lagrange basis:

$$h_i(X) = \prod_{s=1, s\neq i}^{N} \left(\frac{X - X_s}{X_i - X_s}\right) \quad \text{for} \quad i = 1, \ldots, N. \tag{2.5}$$

Similary, using the fluxes at $(N+1)$ flux points, a $(p+1)$-degree polynomial can be built using the following Lagrange basis:

$$l_{i+1/2}(X) = \prod_{s=0, s\neq i}^{N} \left(\frac{X - X_{s+1/2}}{X_{i+1/2} - X_{s+1/2}} \right) \quad \text{for} \quad i = 0, 1, \ldots, N. \tag{2.6}$$

2.2 Global Algorithm

The unsteady update of the solution in solution points Eq. (2.1) follows several steps:

1. Define solution and flux points in each segment of the mesh and initialise the solution at solution points.

   At each time step:

2. Extrapolate the solution Q at flux points using the $p-th$ order polynomial built with known solution values Q at solution points, the solution Q_f is obtained as follows:

   $$Q_f|_{s+1/2} = \sum_{i=1}^{N} Q_i h_i(X_{s+1/2}) \quad \text{for} \quad s \in \{0, 1, ..., N\},$$

   where $N = p + 1$ and Q_i is the solution at solution point i.

3. Compute the flux $\mathcal{F}$ at intern flux points.

4. Compute the flux $\bar{\mathcal{F}}$ at interface of each segment end point using a Riemann solver.

5. Differentiate the flux polynomial at solution points in order to obtain $\frac{\partial \bar{\mathcal{F}}}{\partial x}$:

   $$\left.\frac{\partial \bar{\mathcal{F}}}{\partial x}\right|_s = \sum_{i=0}^{N} \bar{\mathcal{F}}_{i+1/2} l'_{i+1/2}(X_s) \quad \text{for} \quad s \in \{1, ..., N\},$$

   where $\bar{\mathcal{F}}_{i+1/2}$ is the flux at flux point $i + \frac{1}{2}$.

6. Compute of the solution Q at the next time step using $\frac{\partial \bar{\mathcal{F}}}{\partial x}$ values at the solution points and a time integration algorithm.

2.3 Diffusion scheme

To solve full Navier-Stokes equations, the indroduction of a viscous term is needed. The treatment follows the one for the diffusion term of the Navier-Stokes equations and for the sake of clarity, the diffusion scheme is described below. It is based on a pure centred approach, as in [16, 17, 18, 8, 9].

The method to compute any diffusion term with viscosity is similar to the method to compute the advection term. The solution Q at the solution points is extrapolated to the flux points Q_f (FIG. 2.8) and polynomials are not continuous at segment end points.

As introduced above, a centred scheme is considered for the diffusion term and at each interface, the interface value is simply the mean average between values from left and right

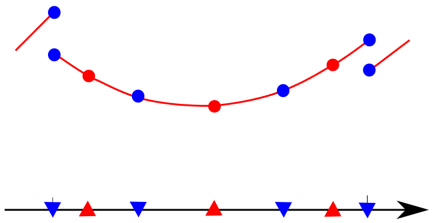

Figure 2.8: Step 1 : Extrapolation of solution (•) at the flux points (▼)

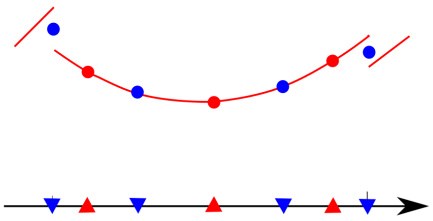

Figure 2.9: Step 2 : Computation of solution (•) at interfaces by mean average

extrapolations: let $\bar{Q}_f$ represent the averaged solution, Fig. 2.9. Using all data at the flux points, it is possible to define an interpolation polynomial from which derivatives are defined at solution points Fig. 2.10. As before, gradient polynomials are built from values at the solution points Fig. 2.11 and the polynomial of ∇Q is known.

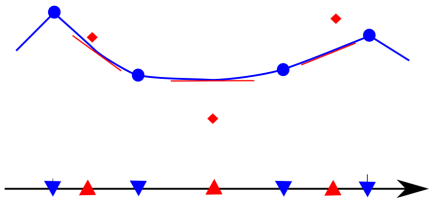

Figure 2.10: Step 3 : Building the continuous polynomial (blue) at flux points and computing the derivative (♦) at solution points

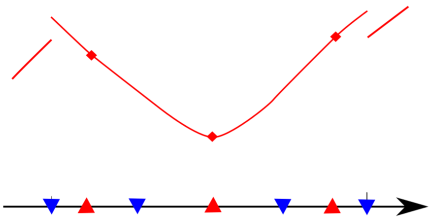

Figure 2.11: Step 4 : Defining the derivative (red) at solution points

∇Q at the flux points is obtained by an interpolation from the gradient polynomial built with gradients are solution points denoted $\nabla \bar{Q}$ (Fig. 2.12).

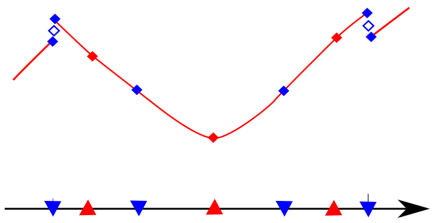

Figure 2.12: Step 5 : Extrapolation of the derivative (⧫) at the flux points

Finally, the interface gradient is defined as the mean of left and right gradients: a continuous gradient polynomial is built. This polynomial can be differentiated in the solution points to give the increment due to the diffusion term.

2.4 Extension to 2D and 3D

This section is devoted to the extension of the SD method to flows in 2 and 3 dimensions. To do so, the solution and flux point locations are given and the second key point concerns the isoparametric transformation to define those locations for any cell. Finally, equations inside the isoparametric elements are derived from the original ones.

2.4.1 Isoparametric transformation

In the mesh, volume sizes can be different and therefore it can be difficult to place solution and flux points. An isoparametric transformation is considered to transform the initial mesh element in the physical domain Ω_ϕ into a square element $\Omega_{ref} = [0,1]^d$ in the reference domain. An example of a 2D case is represented in FIG. 2.13

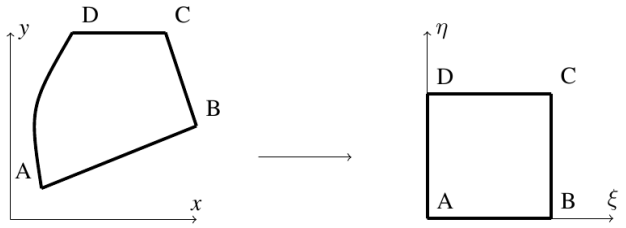

Figure 2.13: Transformation of a quadrilateral cell into the standard cubic element

In 3D, all elements are transformed from the physical domain $(x, y, z) \in \Omega_\phi$ into a standard cubic element $(\xi, \eta, \zeta) \in \Omega_{ref} = [0,1]^3$. The transformation can be written as:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \sum_{i=1}^{K} M_i(\xi, \eta, \zeta) \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}, \tag{2.7}$$

where K is the number of points used to define the physical element, (x_i, y_i, z_i) are the Cartesian coordinates of those points, and $M_i(\xi, \eta, \zeta)$ are the shape functions. For the transformation given in Eq. (2.7), the Jacobian matrix $\mathcal{J}$ takes the following form:

$$\mathcal{J} = \frac{\partial(x, y, z)}{\partial(\xi, \eta, \zeta)} = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \zeta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \zeta} \end{pmatrix}. \tag{2.8}$$

Its determinant is:

$$|\mathcal{J}| = \frac{\partial x}{\partial \xi}\left(\frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \zeta} - \frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \eta}\right) - \frac{\partial y}{\partial \xi}\left(\frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial z}{\partial \eta}\right) + \frac{\partial z}{\partial \xi}\left(\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \eta}\right). \tag{2.9}$$

For a non-singular transformation, its inverse transformation must also exist, and the Jacobian matrices are related to each other according to:

$$\mathcal{J}^{-1} = \frac{\partial(\xi, \eta, \zeta)}{\partial(x, y, z)} = \begin{pmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} & \frac{\partial \xi}{\partial z} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} & \frac{\partial \eta}{\partial z} \\ \frac{\partial \zeta}{\partial x} & \frac{\partial \zeta}{\partial y} & \frac{\partial \zeta}{\partial z} \end{pmatrix}. \tag{2.10}$$

Let Q^{ϕ} be the solution in the physical domain and Q^{ref} be the solution in the reference domain. The hyperbolic equation in the physical domain is:

$$\frac{\partial Q^{\phi}}{\partial t} + \nabla \cdot \mathcal{F}(Q^{\phi}) = 0. \tag{2.11}$$

Indeed, as the transformation changes the solution values, the scalar quantities of the physical solution vector Q and those of the reference solution vector Q^{ref} are linked by an expression that depends on the Jacobian determinant: $Q^{ref} = |\mathcal{J}|Q^{\phi}$, thus the equation becomes

$$\begin{aligned} &|\mathcal{J}|\frac{\partial Q^{\phi}}{\partial t} + |\mathcal{J}|\nabla \cdot \mathcal{F}(Q^{\phi}) = 0 \\ \Leftrightarrow &\frac{\partial Q^{ref}}{\partial t} + |\mathcal{J}|\nabla \cdot \mathcal{F}(Q^{\phi}) = 0. \end{aligned} \tag{2.12}$$

Since:

$$\nabla \cdot \mathcal{F}(Q^{\phi}) = \frac{\partial}{\partial x} f(Q^{\phi}) + \frac{\partial}{\partial y} g(Q^{\phi}) + \frac{\partial}{\partial z} h(Q^{\phi}) \tag{2.13}$$

and

$$\begin{aligned} \frac{\partial f}{\partial x} &= \frac{\partial \xi}{\partial x}\frac{\partial f}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial f}{\partial \eta} + \frac{\partial \zeta}{\partial x}\frac{\partial f}{\partial \zeta}, \\ \frac{\partial g}{\partial y} &= \frac{\partial \xi}{\partial y}\frac{\partial g}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial g}{\partial \eta} + \frac{\partial \zeta}{\partial y}\frac{\partial g}{\partial \zeta}, \\ \frac{\partial h}{\partial z} &= \frac{\partial \xi}{\partial z}\frac{\partial h}{\partial \xi} + \frac{\partial \eta}{\partial z}\frac{\partial h}{\partial \eta} + \frac{\partial \zeta}{\partial z}\frac{\partial h}{\partial \zeta}, \end{aligned} \tag{2.14}$$

it comes:

$$\begin{aligned} \tilde{f}(Q^{\phi}) &= \left(\frac{\partial \xi}{\partial x} f(Q^{\phi}) + \frac{\partial \xi}{\partial y} g(Q^{\phi}) + \frac{\partial \xi}{\partial z} h(Q^{\phi})\right)|\mathcal{J}|, \\ \tilde{g}(Q^{\phi}) &= \left(\frac{\partial \eta}{\partial x} f(Q^{\phi}) + \frac{\partial \eta}{\partial y} g(Q^{\phi}) + \frac{\partial \eta}{\partial z} h(Q^{\phi})\right)|\mathcal{J}|, \\ \tilde{h}(Q^{\phi}) &= \left(\frac{\partial \zeta}{\partial x} f(Q^{\phi}) + \frac{\partial \zeta}{\partial y} g(Q^{\phi}) + \frac{\partial \zeta}{\partial z} h(Q^{\phi})\right)|\mathcal{J}|. \end{aligned} \tag{2.15}$$

$$\Leftrightarrow \begin{pmatrix} \tilde{f} \\ \tilde{g} \\ \tilde{h} \end{pmatrix} = |\mathcal{J}|\mathcal{J}^{-1} \begin{pmatrix} f \\ g \\ h \end{pmatrix}. \tag{2.16}$$

Then, the hyperbolic equation becomes:

$$\frac{\partial Q^{ref}}{\partial t} + \frac{\partial \tilde{f}}{\partial \xi}(Q^{\phi}) + \frac{\partial \tilde{g}}{\partial \eta}(Q^{\phi}) + \frac{\partial \tilde{h}}{\partial \zeta}(Q^{\phi}) = 0. \tag{2.17}$$

2.4.2 Solution points and flux points positions

As in 1D, the solution points are chosen as the Gauss points (Eq. 2.2) and the flux points are chosen as the roots of Legendre polynomials plus two end points. There are $(p+1)^d$ solution points and $(p+2)(p+1)^{d-1}$ flux points, where d is the space dimension. For both 2 and 3D cells, the treatment is performed direction per direction and mathematically, this approach is simply defined as a tensor product, direction per direction. For the solution polynomial, it becomes:

$$(h \otimes h \otimes h)_{i,j,k}(\xi,\eta,\zeta) = h_i(\xi)h_j(\eta)h_k(\zeta)$$

where (ξ,η,ζ) is the location of a solution point in the (i,j,k) directions.

Regarding the flux points, three basis polynomials are needed because there is one for each directional flux point. For $i-$, $j-$ and $k-$ directional flux points, the chosen basis polynomials are given by:

$$\begin{aligned}
(l \otimes h \otimes h)_{i+\frac{1}{2},j,k}(\xi,\eta,\zeta) &= l_{i+\frac{1}{2}}(\xi)h_j(\eta)h_k(\zeta),\\
(h \otimes l \otimes h)_{i,j+\frac{1}{2},k}(\xi,\eta,\zeta) &= l_i(\xi)h_{j+\frac{1}{2}}(\eta)h_k(\zeta),\\
(h \otimes h \otimes l)_{i,j,k+\frac{1}{2}}(\xi,\eta,\zeta) &= l_i(\xi)h_j(\eta)h_{k+\frac{1}{2}}(\zeta).
\end{aligned}$$

An example of the position of solution points and flux points is presented in Fig. 2.14 for $p=3$ and in Fig. 2.15 for $p=2$.

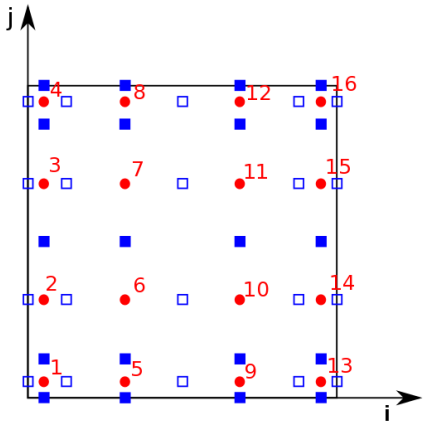

Figure 2.14: Position of solution points (•), i-directional flux points (□) and j-directional flux points (■) for $p=3$

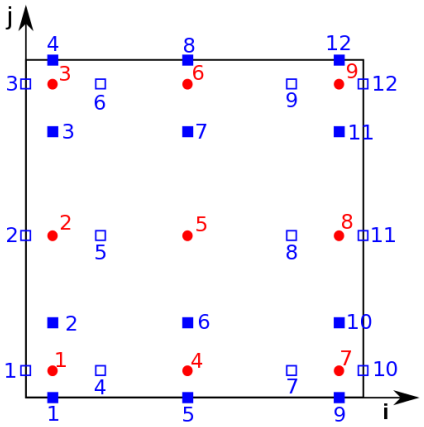

Figure 2.15: Position of solution points and flux points for $p=2$

2.4.3 Algorithm

For computing an approximated solution of the hyperbolic equation in 3D, the SD method is split into several steps:

1. Define solution and flux points in the reference element and introduce the solution.

2. The solution Q^{ϕ} in the physical domain is converted into solution Q^{ref} in the reference domain at solution points: $Q^{ref} = |\mathcal{J}|Q^{\phi}$

   At each time step:

3. Extrapolate the solution Q^{ref} at flux points using the $p-th$ order polynomial built with known solution values Q^{ref} at solution points, leading to the solution Q_f^{ref}. Introducing $N = p+1$, it comes:

$$Q_f^{ref}(\xi,\eta,\zeta) = \sum_{k=1}^{N}\sum_{j=1}^{N}\sum_{i=1}^{N} Q_{i,j,k}^{ref} h_i(\xi)h_j(\eta)h_k(\zeta),$$

   where $Q_{i,j,k}^{ref}$ is the solution at solution point (i,j,k) in the reference domain.

4. The solution Q_f^{ref} in the reference domain is converted into solution Q_f^{ϕ} in the physical domain at flux points: $Q_f^{\phi} = \frac{Q_f^{ref}}{|\mathcal{J}|}$

5. Compute the flux $\bar{f}$, $\bar{g}$ and $\bar{h}$ at intern flux points in the physical domain.

6. Compute the flux $\bar{f}$, $\bar{g}$ and $\bar{h}$ at interfaces of each element using a Riemann solver in the physical domain.

7. The flux $\bar{f}$, $\bar{g}$ and $\bar{h}$ in the physical domain is converted into flux $\tilde{f}$, $\tilde{g}$ and $\tilde{h}$ in the reference domain:

$$\begin{aligned}
\tilde{f} &= \left(\frac{\partial\xi}{\partial x}f + \frac{\partial\xi}{\partial y}g + \frac{\partial\xi}{\partial z}h\right)|\mathcal{J}|,\\
\tilde{g} &= \left(\frac{\partial\eta}{\partial x}f + \frac{\partial\eta}{\partial y}g + \frac{\partial\eta}{\partial z}h\right)|\mathcal{J}|,\\
\tilde{h} &= \left(\frac{\partial\zeta}{\partial x}f + \frac{\partial\zeta}{\partial y}g + \frac{\partial\zeta}{\partial z}h\right)|\mathcal{J}|.
\end{aligned} \tag{2.18}$$

8. Differentiate the flux polynomial in the solution points in order to obtain $\frac{\partial\tilde{f}}{\partial\xi}$, $\frac{\partial\tilde{g}}{\partial\eta}$ and $\frac{\partial\tilde{h}}{\partial\zeta}$:

$$\begin{aligned}
\frac{\partial\tilde{f}}{\partial\xi}(\xi,\eta,\zeta) &= \sum_{r=0}^{N}\tilde{f}_{r+1/2,j,k}l'_{r+1/2}(\xi)h_j(\eta)h_k(\zeta),\\
\frac{\partial\tilde{g}}{\partial\eta}(\xi,\eta,\zeta) &= \sum_{r=0}^{N}\tilde{g}_{i,r+1/2,k}l'_{r+1/2}(\eta)h_i(\xi)h_k(\zeta),\\
\frac{\partial\tilde{h}}{\partial\zeta}(\xi,\eta,\zeta) &= \sum_{r=0}^{N}\tilde{h}_{i,j,r+1/2}l'_{r+1/2}(\zeta)h_i(\xi)h_j(\eta).
\end{aligned}$$

9. Compute the solution Q^{ref} at the next time step thanks to $\frac{\partial\tilde{f}}{\partial\xi}$, $\frac{\partial\tilde{g}}{\partial\eta}$ and $\frac{\partial\tilde{h}}{\partial\zeta}$ values at the solution points and time integration algorithm.

Output: the solution Q^{ref} in the reference domain is converted into solution in the physical domain at solution points: $Q^{\phi} = \frac{Q^{ref}}{|\mathcal{J}|}$.

Chapter 3

Numerical results on a convected 2D isentropic vortex

This chapter is dedicated to the analysis of numerical results obtained for a simple case: the 2D convection of an isentropic vortex. This case is chosen for its simplicity and the possibility to quantify theoretically the error with respect to the initial "exact" solution. Indeed, as solution of the Euler equations, the isentropic vortex has to remain unchanged during the simulation. Many simulations of the 2D vortex convection have already been performed with JAGUAR and compared to the results obtained with other CFD codes. Here, the goal is to implement and test different methods to improve the efficiency and the accuracy of the simulation.

3.1 Vortex initialization and description

An isentropic 2D convected vortex can be generated using the following function

$$\psi(x,y) = \Gamma \exp\left(-\frac{(x-x_c)^2+(y-y_c)^2}{2R_c^2}\right), \tag{3.1}$$

where Γ is the circulation, R_c is the radius and (x_c, y_c) are the coordinates of the vortex center. Velocity components and pressure can be obtained from:

$$u = U_0 + \frac{\partial \psi}{\partial y}, v = -\frac{\partial \psi}{\partial x} \text{ and } P - P_0 = -\frac{\rho\Gamma^2}{2R_c^2}\exp\left(-\frac{(x-x_c)^2+(y-y_c)^2}{R_c^2}\right).$$

U_0 and P_0 are respectively the transport velocity and the ambient pressure. Simulations are performed in a $0.1m$ biperiodic 2D box. The vortex is defined with a constant circulation $\Gamma = 34.728$, $(x_c, y_c) = (0.05, 0.05)$ and $R_c = 0.005$ (FIG. 3.1). The flow is initialized with a constant density $\rho_0 = 1.1608 kg.m^{-3}$, a constant temperature $T_0 = 300K$, a constant pressure $P_0 = 10^5 Pa$ and a constant transport Mach velocity M. To avoid problems on the boundaries, the function defined in Eq. 3.1 is truncated as soon as the point (x, y) leaves the circle of diameter $8R_c$.

In the following, attention will be paid on the L2 norm error of the density (Eq. 3.2) and, particularly, of the pressure (Eq. 3.3) to compare our results with previous simulations:

$$\Delta\rho = \int_V (\rho_{num} - \rho_{exact})^2 \, dV \tag{3.2}$$

$$\Delta P = \int_V (P_{num} - P_{exact})^2 \, dV \tag{3.3}$$

For the numerical computation of the integrals, a high order quadrature rule in agreement with the polynomial degree choice is considered. As a consequence, the numerical error is evaluated with a procedure in agreement with the polynomial representation of quantities. In particular, a standard integration procedure like the one considered in Finite Volume approximation leads to results of poor quality.

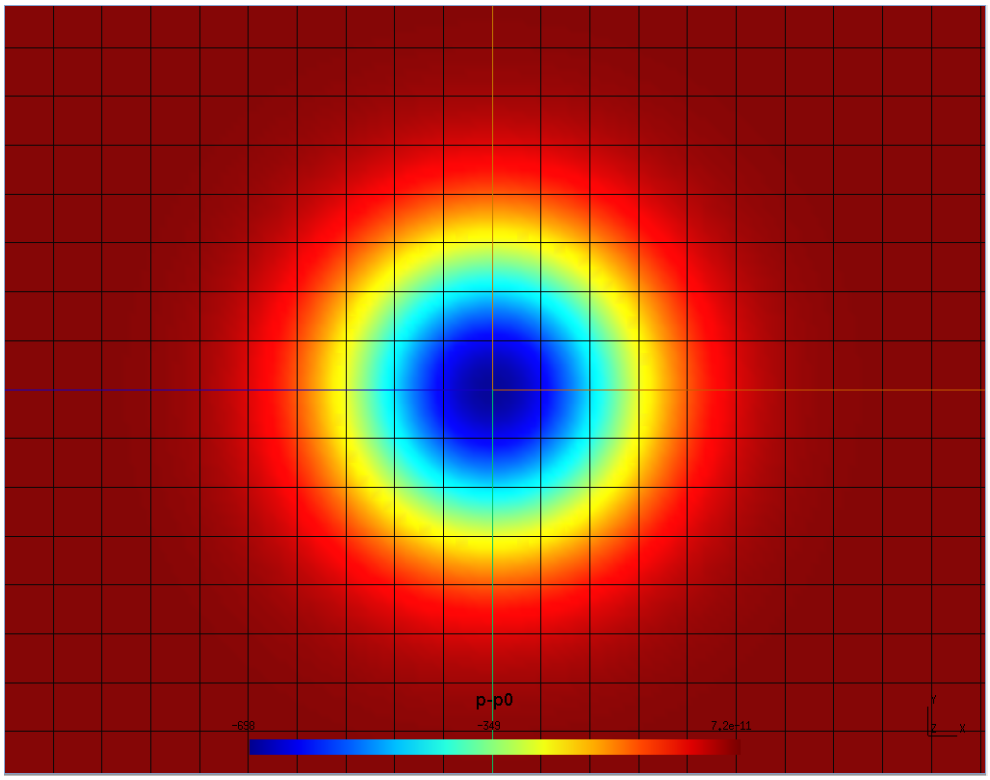

Figure 3.1: Initialisation of a COVO

3.2 Modification of the Riemann solver

The SD method is based on accounting explicitly for the discontinuity between two cells by the use of a Riemann solver. The choice of a Riemann solver can have an influence on the overall quality of the solution. Initially, the Roe's approximated Riemann solver was implemented in JAGUAR [13].

JAGUAR has been compared with several CFD codes for the convection of the vortex and at different transport Mach number M. In particular, it is well known that the Roe's scheme is not accurate for low Mach number transport ($M \lessapprox 0.1$) and its use in such a configuration needs a low-speed preconditioning technique. Such a technique was not available in JAGUAR and the final results were not as accurate as the results obtained with other CFD codes. This is why two new Riemann solvers - AUSM+up [6] and SLAU [15]- were implemented in JAGUAR and tested. Moreover, the simple Rusanov scheme [14, 5, 19] is generally considered in the literature and it has also been considered to give a reference on solution accuracy.

3.2.1 Convected vortex at M=0.5

3.2.1.1 Comparison of solvers

Here, a comparison between the results obtained with the four solvers implemented in JAGUAR is realized. Simulations were performed on 16 processors, with a 6^{th} order Runge-Kutta time integration (RK6SD [2]), 4^{th} order polynomials modelling the solution in each cell ($p = 4$), with a very fine mesh (FIG. 3.3) and at two CFL numbers: 0.8 and 0.2. The simulations are prepared to end after 50 advective times, an advective time being the time the vortex needs to cross the periodic box and come back to its initial position. In the simulations presented here, $t_f = 0.028795s$.

Riemann solver	$\Delta\rho$	ΔP	Total CPU Time(s)
Roe	$3.10\ 10^{-7}$	$3.74\ 10^{-2}$	12312
Rusanov	$3.06\ 10^{-7}$	$3.69\ 10^{-2}$	11472
AUSM+up	$3.13\ 10^{-7}$	$3.78\ 10^{-2}$	12048
SLAU	$3.10\ 10^{-7}$	$3.74\ 10^{-2}$	11712

Table 3.1: Performance of the different solvers with CFL=0.8

Riemann solver	$\Delta\rho$	ΔP	Total CPU Time(s)
Roe	$3.73\ 10^{-8}$	$4.49\ 10^{-3}$	48272
Rusanov	$2.89\ 10^{-8}$	$3.48\ 10^{-3}$	45696
AUSM+up	$4.37\ 10^{-8}$	$5.23\ 10^{-3}$	47344
SLAU	$3.75\ 10^{-8}$	$4.50\ 10^{-3}$	47016

Table 3.2: Performance of the different solvers with CFL=0.2

Results in TABLE 3.1 and TABLE 3.2 show that, in this configuration, the four solvers have very similar behaviors, with errors being of the same order. Nevertheless, it can be underligned that Rusanov's solver is the fastest one (which is not really surprising as it has the lower number of operations per degree of freedom among the four) closely followed by SLAU solver.

SLAU (Simple Low-dissipation AUSM) was implemented to correct the bad behavior of Roe and AUSM-family solvers at low Mach numbers. It has shown accuracy, robustness and efficiency for computations of wide-ranging Mach numbers, especially for low Mach number flows.

For the following simulations, attention will be focused on Roe and SLAU solvers.

3.2.1.2 Effects of the time integration scheme

In JAGUAR, four time integration schemes are implemented: 4^{th}-order Runge-Kutta (RK4), 4^{th}-order Runge-Kutta with low storage (RK4ls), 6^{th}-order Runge-Kutta (RK6SD) [2] and 6^{th}-order Runge-Kutta with low dissipation and low dispersion (RK6ldld) [1]. The goal here is to observe the influence of the time integration scheme on the results obtained with simulations ran by JAGUAR. For the considered simulations, $p = 4$ and the simulations are performed on a parallel supercomputer with 16 cores on the very fine mesh. Computations are stopped after 50 advective times.

Riemann solver	Time integration	CFL	$\Delta\rho$	ΔP	Total CPU Time(s)
Roe	RK4	0.8	Negative Time Step		
		0.2	Negative Time Step		
	RK4ls	0.8	Negative Time Step		
		0.5	$9.04\ 10^{-8}$	$1.09\ 10^{-2}$	13104
		0.2	$3.41\ 10^{-8}$	$4.11\ 10^{-3}$	32616
	RK6SD	0.8	$3.10\ 10^{-7}$	$3.74\ 10^{-2}$	12312
		0.2	$3.73\ 10^{-8}$	$4.49\ 10^{-3}$	48272
	RK6ldld	0.8	$1.95\ 10^{-4}$	$2.35\ 10^{2}$	12096
		0.4	$8.89\ 10^{-8}$	$1.07\ 10^{-2}$	24192
		0.2	$3.73\ 10^{-8}$	$4.49\ 10^{-3}$	48208
SLAU	RK4	0.8	NaN (5 iterations)		
		0.2	NaN (5 iterations)		
	RK4ls	0.8	$2.07\ 10^{-2}$	$1.66\ 10^{3}$	9072
		0.5	$8.99\ 10^{-8}$	$1.08\ 10^{-2}$	12640
		0.2	NaN (5 iterations)		
	RK6SD	0.8	$3.10\ 10^{-7}$	$3.74\ 10^{-2}$	11712
		0.2	$3.75\ 10^{-8}$	$4.50\ 10^{-3}$	47016
	RK6ldld	0.8	$3.10\ 10^{-7}$	$3.74\ 10^{-2}$	11888
		0.2	$3.75\ 10^{-8}$	$4.50\ 10^{-3}$	46896

Table 3.3: Influence of the time integration scheme and the CFL number associated on the total CPU time and the L2 error of density and pressure

One can see in Table 3.3 that RK4 scheme is not working well, leading to negative time steps with Roe's solvers or to NaNs with SLAU solver. For the RK4ls scheme, despite it is faster by about a third than the 6^{th} order schemes (which comes from the fact that there is "only" four steps at each time step instead of six for the 6^{th} order schemes) and that it gives the best result of all these tests with Roe's solver at $CFL = 0.5$, results are strongly dependent of the CFL number and the Riemann solver chosen. This is a real drawback compared to the wide range of CFL number that leads to a stable and accurate result with the 6^{th} order schemes. Eventually, one can see that RK6ldld does not work well at $CFL = 0.8$ with Roe solver. This lack of accuracy does not appear with RK6SD scheme. Both schemes having similar performances in the other cases, RK6SD scheme appears to be a little better than RK6ldld. For the next simulations, and unless it is specified, the RK6SD scheme will be used.

3.2.1.3 Evolution of the performance with the mesh

The objective here is to compare the accuracy and the cost of the simulation ran with four different meshes (for JAGUAR):

- M1: 8×8 grid (Fig. 3.2) on 2 cores,
- M2: 16×16 grid on 2 cores,
- M3: 32×32 grid on 8 cores,
- M4: 64×64 grid (Fig. 3.3) on 16 cores.

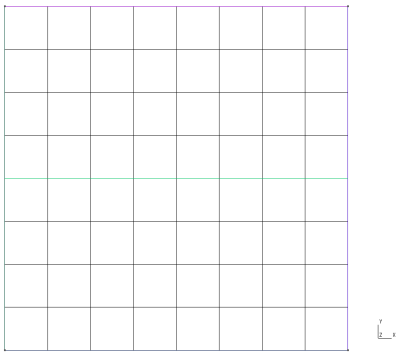

Figure 3.2: Coarse mesh M1

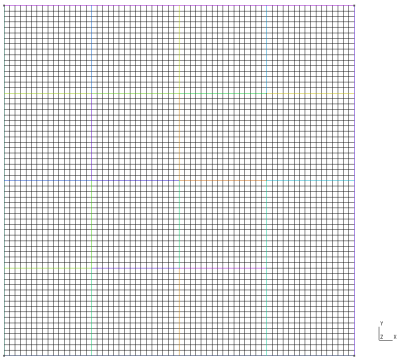

Figure 3.3: Very fine mesh M4

Simulations were already performed with JAGUAR and AVBP (JAGUAR and AVBP meshes are different). Fig. 3.4 shows that JAGUAR and AVBP have close CPU costs and that growing the polynomial order in JAGUAR makes the simulation more accurate. The gain in accuracy reaches one order of magnitude for JAGUAR p4 compared to AVBP on the M4 mesh.

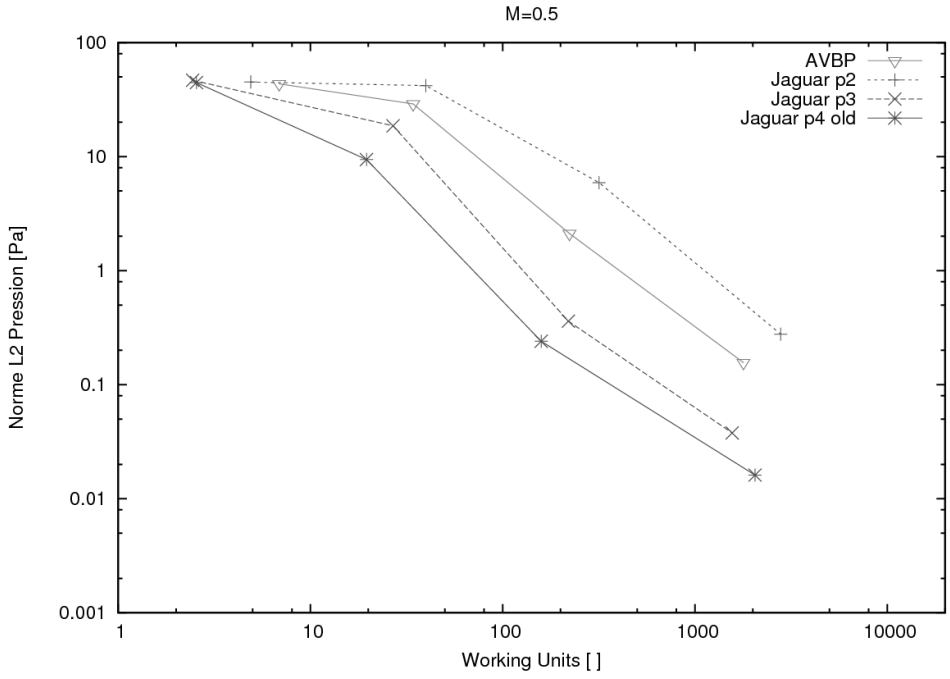

Figure 3.4: Previous results on the evolution of accuracy and CPU time function of the mesh for different CFD codes at Mach=0.5

New simulations were run with JAGUAR (4^{th} order polynomial). Results are presented and compared to the previous ones in Fig. 3.5. It can be seen that at $CFL = 0.8$, Roe and SLAU solvers have the exact same performances. For the three coarser meshes, those performances also match the previous results. Differences appear only for the finer mesh, new simulations being faster but less accurate. Moreover, when the CFL is reduced to 0.2, Roe and

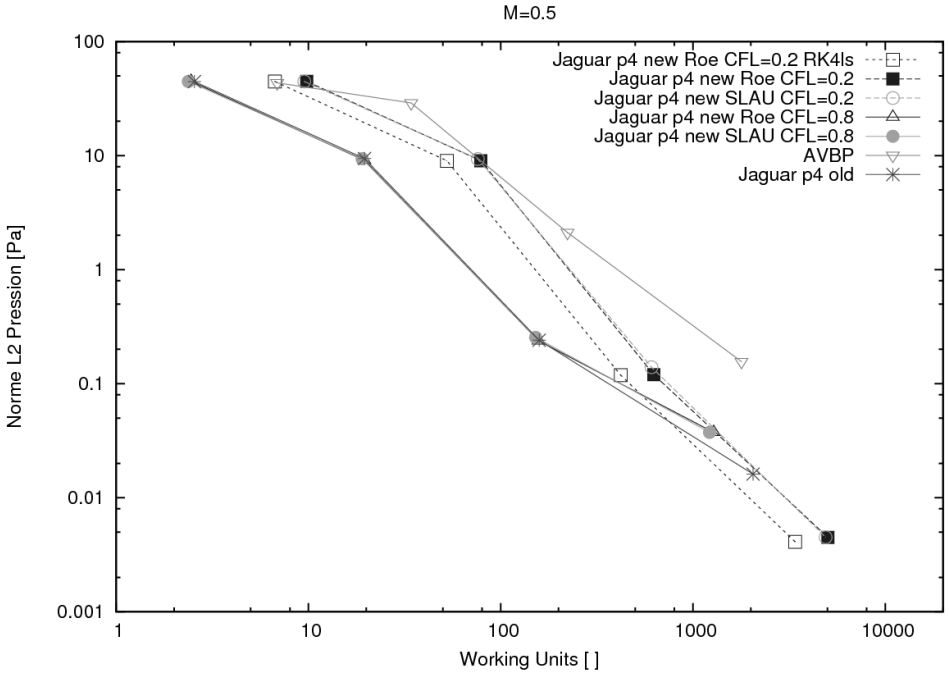

Figure 3.5: Comparison between previous and last results on the evolution of accuracy and CPU time function of the mesh at Mach=0.5

SLAU solvers have, once again, the same performances. It can be noticed that, as expected, for all meshes, CPU time needed to reach the end of the simulation is greatly increased (by a factor around 4). However, gain in accuracy only appears with the finer meshes (M3 and more importantly M4). This can be explained by the fact that, in the coarser meshes, the spatial error is prevailing over the temporal error and reducing the CFL does not affect the overall error. On the contrary, with finer meshes, time-related error is overwhelming and reducing the time step allows to greatly improve the overall error (almost a factor 10 with M4).

3.2.2 Convected vortex at M=0.001

Initial conditions are conserved for this new serie of simulations except for the Mach number, reduced to 0.001 (hence a reduction of the depression in the vortex), and the final time modified for the simulation to end after 10 advective times (here $t_f = 2.8795s$). As for the simulations at $M = 0.5$, previous results were already released and results of the new simulations with Roe and SLAU solvers will be compared to those previous ones.

First, it can be noticed in Fig. 3.6 that AVBP became really faster than JAGUAR with a better or, at least, equivalent accuracy. Concerning the newer simulations, results for Roe's solver match the curve already obtained. For the SLAU solver, results show that the same CPU time as Roe's solver is needed to reach t_f but the accuracy of the result is much better, with a gain of one order of magnitude. This was expected since SLAU solver was built as a correction to the bad behaviour of Roe's solver at low Mach numbers.

However, even if it became more accurate, JAGUAR code with SLAU solver is still far from catching the speed of AVBP at $M = 0.001$. In order to correct that, a new method, already

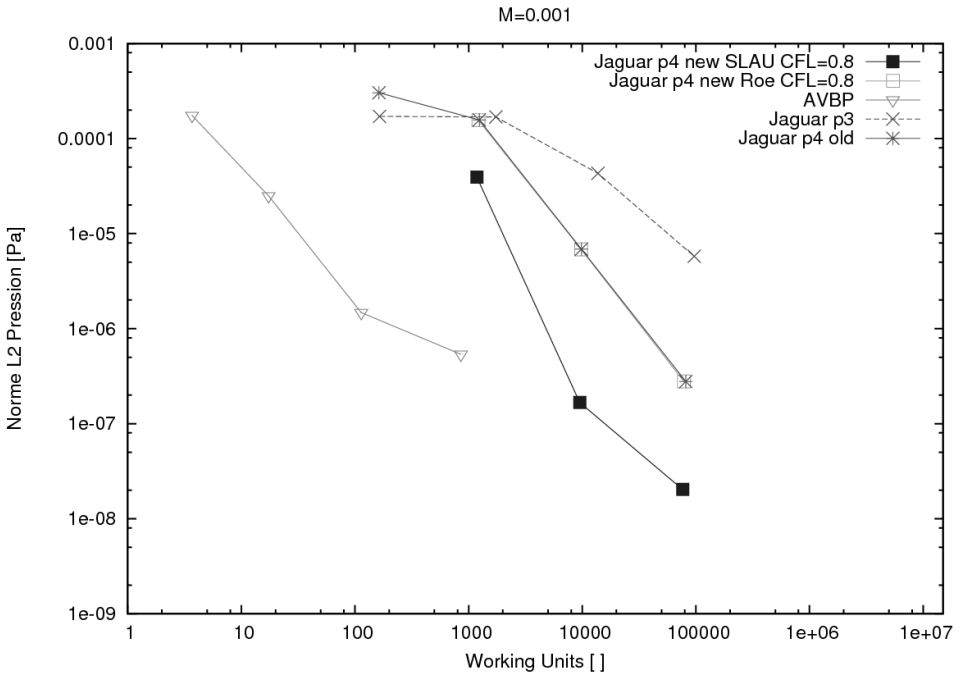

Figure 3.6: Comparison between previous and last results on the evolution of accuracy and CPU time function of the mesh at Mach=0.001

present in AVBP code, has been implemented in JAGUAR: the cryogenic approach.

3.3 Cryogenic approach

The main idea of this approach is to keep the conservative fields and artificially modify the temperature and the pression (at constant density) inside the flow in order to reduce the value of the sound speed. This leads to the increase of the apparent Mach number hence it reduces the CPU time needed to end the simulation [11, 10].

3.3.1 Principle

Transforming a standard simulation into a cryogenic one needs a few modifications:
Before starting the simulation:

1. Choose the acoustic Mach M_a of the cryogenic simulation (with $M_a \leq 0.3$ to respect the constant density hypothesis)

2. Calculate ρ and u with the real condition inputs ($P = 10^5 Pa$, $T = 300K$ and $M = 10^{-3}$ in our current case).

3. Once u and ρ are known, determine T_{cryo}^{input} and P_{cryo}^{input} using the two relations:

$$T_{cryo}^{input} = \frac{1}{\gamma R}\left(\frac{u}{M_a}\right)^2 \tag{3.4}$$

$$P_{cryo}^{input} = \rho R T_{cryo} = \frac{\rho}{\gamma}\left(\frac{u}{M_a}\right)^2 \tag{3.5}$$

4. Compute the following variable change to be in cryogenic conditions:

$$\begin{cases} T_{cryo} &= T_{real} - T_{real}^{input} + T_{cryo}^{input} \\ P_{cryo} &= P_{real} - P_{real}^{input} + P_{cryo}^{input} \end{cases} \tag{3.6}$$

   Once the simulation reaches its end, the objective is to get back to real conditions to post-treat the simulation.

5. Doing so is easy. It is only needed to invert Sys. (3.6):

$$\begin{cases} T_{real} &= T_{cryo} + T_{real}^{input} - T_{cryo}^{input} \\ P_{real} &= P_{cryo} + P_{real}^{input} - P_{cryo}^{input} \end{cases} \tag{3.7}$$

6. Eventually, we want to compute the error ΔP.

   Concerning the calculation of the pressure error $\Delta P = \int_V \left(P_{num}^{real} - P_{exact}^{real}\right)^2 dV$, we have the two following relations:

$$\begin{cases} P_{num}^{cryo} &= P_{num}^{real} - P_{mean} + P_{cryo}^{input} \\ P_{exact}^{cryo} &= P_{exact}^{real} - P_{mean} + P_{cryo}^{input} \end{cases} \tag{3.8}$$

   Two solutions to compute ΔP are then available:

   a. Use the first relation of Sys. (3.8) to calculate P_{num}^{real} and compute the error in the real conditions

   b. By differentiating term by term Sys. 3.8, we can observe that

$$P_{num}^{cryo} - P_{exact}^{cryo} = P_{num}^{real} - P_{exact}^{real}. \tag{3.9}$$

      This means that computing the error directly in cryogenic conditions is exactly the same as getting back to real conditions and then computing the error.

   It was chosen to implement the second solution in JAGUAR.

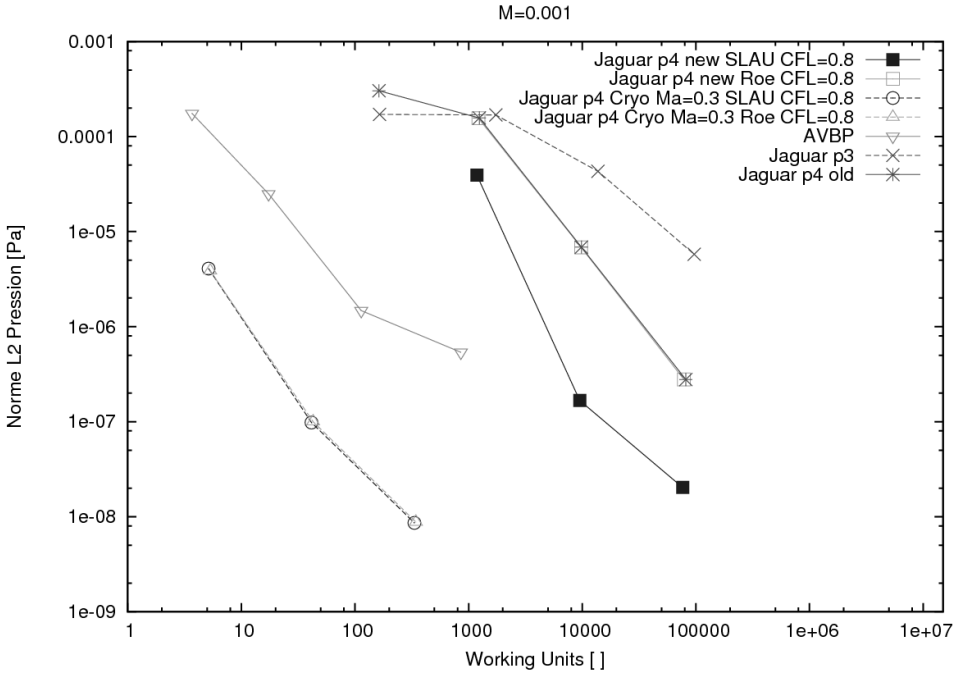

Figure 3.7: Comparison between cryogenic and standard simulations at Mach=0.001

3.3.2 Results

3.3.2.1 M=0.001

For the results presented in the following, simulations were run with Roe and SLAU solvers at $CFL = 0.8$ and at $M_a = 0.3$, hence we had $T_{cryo} = 0.003333K$ and $P_{cryo} = 1.111111N.m^{-2}$.

Fig. 3.7 summarizes the results obtained.

What can be underligned first is that, in cryogenic conditions, Roe and SLAU solvers have equivalent performances. This can be explained by the fact that in cryogenic conditions, the apparent stream Mach number is $M_a = 0.3$. This means the apparent Mach number of the flow in the cryogenic simulation is high enough for the Roe solver to be in its optimal performance domain, making the SLAU amelioration less useful. Moreover, as it was expected, cryogenic simulations are much faster than standard simulations (up to 250 times faster) and we can even observe a gain in accuracy. This evolution makes JAGUAR the fastest and most accurate CFD code (developed at CERFACS) on this particular convected vortex application.

3.3.2.2 M=0.05

Here, Fig. 3.8 presents the results of simulations ran with SLAU solver in standard conditions and in cryogenic conditions, for which acoustic Mach number were respectively $M_a = 0.14$ and $M_a = 0.3$.

Again, it can be noticed that SLAU solver is more accurate than and as fast as Roe solver for low intermediate Mach numbers, even if the gain is quite moderated, compared with simulations at $M_a = 0.001$. Using the cryogenic approach reduces the CPU time, but not as much as it did at $M = 0.001$, as the difference between real and cryogenic Mach number is

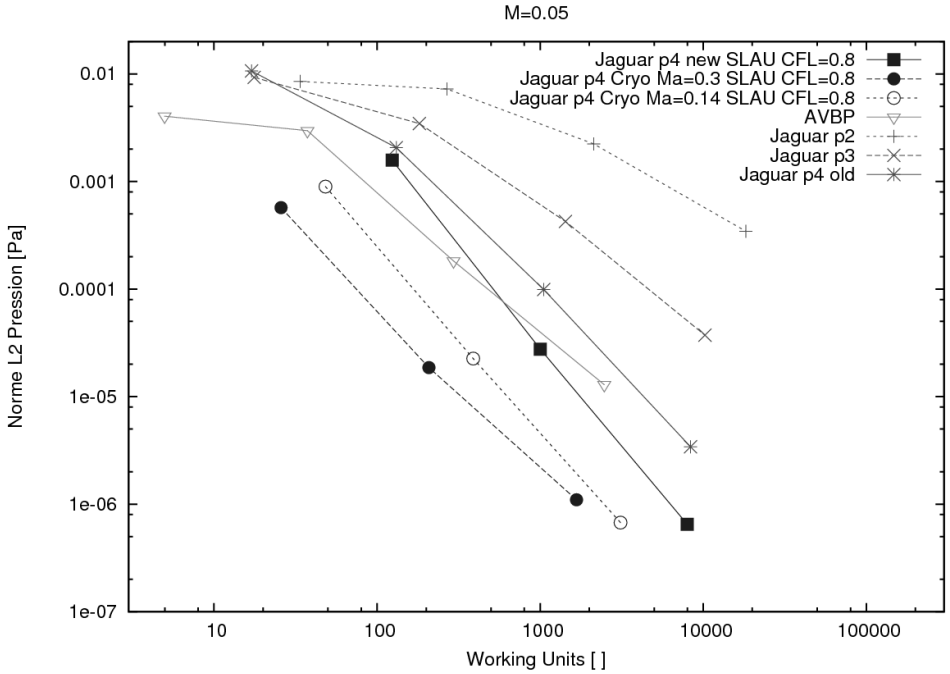

Figure 3.8: Comparison between cryogenic and standard simulations at Mach=0.05

smaller. However, it allows to catch AVBP CPU time performance using the same approach.

3.4 Conclusions on the convected vortex simulations

It has been seen that JAGUAR had already good performances at high subsonic Mach numbers ($M \gtrapprox 0.1$). But the lack in performance of Roe solver and the better CPU time of AVBP at low Mach numbers constituted two enhancements for the performances of JAGUAR. In order to increase those performances, several modifications were tested:

- Changing the Riemann solver from Roe to SLAU. This modification brought a noticeable increase in accuracy at low Mach numbers without modifying the time of the simulation,
- Implementation of the cryogenic approach, inspired by AVBP. This modification only has effects on low Mach number simulations ($M \leq 0.3$) and this approach allowed JAGUAR to become much faster and even a more accurate at very low Mach numbers.

The future objectives will be to see if those methods work as well as they did here in other applications.

Chapter 4

Conclusion and perspectives

This internship report deals with unsteady simulations using a new CFD code based on the Spectral Difference (SD) paradigm. The SD method is a high-order approach to solve hyperbolic and elliptic PDEs based on polynomial approximation inside each cell. It is not required to have a continuous representation of the solution inside the whole computational domain and the SD method is a member of discontinuous spectral approaches.

By nature, the SD approach allows data discontinuity at mesh interface and the discontinuity in data is accounted for by an approximated Riemann solver. In JAGUAR, the first Riemann solver available was the Roe's solver. However, this solver is known to lack accuracy at low Mach numbers. Consequently, three other Riemann solvers (Rusanov, AUSM+-up and SLAU) were implemented recently. Simulations of a 2D convected vortex inside a biperiodic box enabled to analyze both accuracy and CPU performance. The results were compared to the ones obtained with Roe solver on the same simulation. It appears that SLAU and Roe solvers have equivalent performances at high Mach number while SLAU solver allows a significant gain in accuracy without increasing the CPU time at low Mach numbers.

Moreover, another axis of development was identified: the reduction of the CPU time. To do so, the cryogenic approach was implemented in JAGAUR. This approach is based on the artificial diminution of the temperature and the pressure at constant density for low Mach number flows ($M < 0.3$). These modifications reduce the speed of sound in the flow and make the apparent Mach number larger, reducing the number of time steps needed to reach the end of the unsteady simulation. Simulations ran on the 2D convected vortex showed great results, the CPU time being reduced by a factor up to 250 (for a configuration with a real Mach number $M = 0.001$ and an apparent cryogenic Mach number $M_a = 0.3$).

During this internship, it was also planed to run and post-treat a JAGUAR 128-processors LES of a plan channel. The simulation ran but lack of time made the post-treatment impossible to be realized before the end of the internship. The post-treatment will then be performed by my supervisor in a close future and will be presented in future papers.

Bibliography

[1] C. Bogey and C. Bailly. A family of low dispersive and low dissipative explicit schemes for flow and noise computations. *Journal of Computational Physics*, 194:194–214, 2004.

[2] N. Villedieu J. Vanharen, G. Puigt, 2013. Private Communications.

[3] A. Jameson. A proof of the stability of the spectral difference method for all orders of accuracy. *Journal of Scientific Computing*, 45:348–358, 2010.

[4] D.A. Kopriva and J.H. Kolias. A conservative staggered-grid Chebyshev multidomain method for compressible flows. *Journal of Computational Physics*, 125(1):244–261, 1996.

[5] P. D. Lax. Weak solutions of non-linear hyperbolic equations and their numerical computation. *Communications in Pure and Applied Mathematics*, VII:159–193, 1954.

[6] M-S. Liou. A sequel to AUSM, part II: AUSM$^+$-up for all speeds. *Journal of Computational Physics*, 214:137–170, 2006.

[7] Y. Liu, M. Vinokur, and Z.J. Wang. Spectral difference method for unstructured grids I: Basic formulation. *Journal of Computational Physics*, 216(2):780–801, 2006.

[8] G. Lodato. High-order schemes for high-fidelity DNS and LES of complex geometries. In *CERFACS conference on High Order Spectral Difference Method*, 2014.

[9] G. Lodato, P. Castonguay, and A. Jameson. Structural wall-modeled LES using a high-order spectral difference scheme for unstructured meshes. *Flow, Turbulence and Combustion*, 92(1-2):579–606, January 2014.

[10] P. J. O'Rourke and F. V. Bracco. Two scaling transformations for the numerical computation of multidimensional unsteady laminar flames. *Journal of Computational Physics*, 33:185–203, 1979.

[11] J. D. Ramshaw, P. J. O'Rourke, and L. R. Stein. Pressure gradient scaling method for fluid flow with nearly uniform pressure. *Journal of Computational Physics*, 58:361–376, 1985.

[12] W.H. Reed and T.R. Hill. Triangular mesh methods for the neutron transport equation. Technical report, Los Alamos National Laboratory, New Mexico, USA, Tech. Report LU-UR-73-279, 1973.

[13] P. L. Roe. Approximate Riemann solvers, parameter vectors and difference schemes. *Journal of Computational Physics*, 43:357–372, 1981.

[14] V. V. Rusanov. Calculation of intersection of non-steady shock waves with obstacles. *Journal of Computational Mathematics and Physics USSR*, 1:267–279, 1961.

[15] E. Shima and K. Kitamura. On new simple low-dissipation scheme of ausm-family for all speeds. In *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition, 5 - 8 January, Orlando, Florida, AIAA Paper 2009-136*, 2009.

[16] Y. Sun, Z.J. Wang, and Y. Liu. High-order multidomain spectral difference method for the Navier-Stokes equations. In *44th AIA Aerospace Sciences Meeting and Exhibit, AIAA Paper 2006-301*, 2006.

[17] Y. Sun, Z.J. Wang, and Y. Liu. High-order multidomain spectral difference method for the Navier-Stokes equations on unstructured hexahedral grids. *Communications in Computational Physics*, 2:310–333, 2007.

[18] Y. Sun, Z.J. Wang, and Y. Liu. Efficient implicit non-linear LU-SGS approach for compressible flow computation using high-order spectral difference method. *Communications in Computational Physics*, 5(2-4):760–778, February 2009.

[19] E.F. Toro. *Riemann solvers and Numerical Methods for Fluid Dynamics - A Practical Introduction*. Springer, 2nd edition edition, 1999.

[20] K. Van den Abeele, C. Lacor, and Z.J. Wang. On the stability and accuracy of the spectral difference method. *Journal of Scientific Computing*, 37:162–188, 2008.

[21] Z.J. Wang. Spectral (finite) volume method for conservation laws on unstructured grids: Basic formulation. *Journal of Computational Physics*, 178(1):210–251, 2002.

[22] Z.J. Wang and Y.Liu. Spectral (finite) volume method for conservation laws on unstructured grids II: Extension to two-dimensional scalar equation. *Journal of Computational Physics*, 179(2):665–697, 2002.