Escola de Primavera de Transição e Turbulência
Blumenau 2020

ABCM Associação Brasileira de Engenharia e Ciências Mecânicas

12th Spring School on Transition and Turbulence
September 21st-25th, 2020, Blumenau, SC, Brazil

EPTT-2020-0116
SIMEX IMPLEMENTATION ON A DNS CODE

Giovanni Belloni Fernandes Braga
Marcello Augusto Faraco de Medeiros
University of São Paulo - USP
gbelloni@usp.br
marcello@sc.usp.br

Sávio Brochini Rodrigues
Federal University of São Carlos - UFSCar
savio.rodrigues@ufscar.br

Abstract. *This study is dedicated to the implementation, verification and measurement of execution speed, using the method of manufactured solutions, of a new temporal integration method of the implicit-explicit kind, the Stepwise-IMEX or SIMEX, in the Aeroacoustics, Transition and Turbulence Group's (GATT) Direct Numerical Simulation (DNS) code. In simulations where $Ma \approx 0.1$, or less, SIMEX should be able to reduce the explicit 4^{th} order Runge-Kutta's (ERK) execution time, which is currently used. A pilot program was created, in Python, with the Burgers' 1D equation, using 4^{th} order compact finite differences, to test from 2^{nd} to 5^{th} order SIMEX methods and the mentioned explicit method. Indeed, in these preliminary tests, SIMEX proved to be faster than ERK only with linear Burgers' equation and only after a certain stiffness level. However, there are indications that some characteristics of the DNS, such as the y axis' greater refinement, not found in the pilot are favorable to SIMEX. Both the decomposition of the ODE and the iterative method for implicit systems and the chosen tableaus pair have a huge impact on the SIMEX efficiency and will be targeted in next stage of DNS code implementation.*

Keywords: *IMEX, SIMEX, Runge-Kutta, Temporal integration.*

1. INTRODUCTION

The current group's (Turbulence, Transition and Aeroacoustic Group - GATT) Direct Numerical Simulation (DNS) code is a reorganization of the previous versions, (Mathias, 2017). Details about its use and validation can be found at (Silva *et al.*, 2010; Bergamo, 2014; Bergamo *et al.*, 2015; Martinez and Medeiros, 2016; Martinez, 2016; Mathias, 2017). In this code the acoustic phenomena are always resolved which allows aeroacoustic analysis for every subsonic Mach number. At low Mach numbers, say $0 < Ma \approx 0.1$, the velocity fields are much slower then the acoustic waves, which in pratice makes the system stiff, thus imposing time-step restrictions for the simulations with explicit integration methods.

Fully implicit methods build up complexity and CPU cost proportionally to the integrated equation's complexity. So it is quite clear that Navier-Stokes equations are not very well suited for these kind of methods at first glance. But, because different terms contribute distinctly to the stiffness, implicit-explicit methods can offer some advantage. This class of methods decomposes the equation in two parts, the part with the stiff terms are integrated implicitly and the remaining part is integrated explicitly. Bigger stable time-steps may reduce execution time as consequence of this approach. The trade-off to be managed is the necessity of keeping the implicit equation simple to solve against the possibility of using bigger time-steps.

The use of implicit and explicit methods in an ODE system dates back to the 1970s. Early studies are (Hofer, 1976), (Crouzeix, 1980) and (Cooper and Sayfy, 1980). A comprehensive review can be found in (Christopher A. Kennedy, 2001; Kennedy and Carpenter, 2003). IMEX methods have been studied with a variety of time-stepping schemes. Here we use singly-diagonally implicit Runge-Kutta with explicit first stage (ESDIRK) schemes (Ascher *et al.*, 1997; Ghosh and Constantinescu, 2016; Christopher A. Kennedy, 2018; Wang *et al.*, 2015). A comprehensive review of ESDIRK schemes can be found in (Christopher A. Kennedy, 2016).

Recently, IMEX methods have been used in compressible fluids simulations such as in atmospheric studies, where the stiffness is associated with terms that generates acoustic waves (Bispen *et al.*, 2017; Zhang *et al.*, 2016b; Colavolpe *et al.*, 2017; Durran and Blossey, 2012; David J. Gardner *et al.*, 2018; Ghosh and Constantinescu, 2016; Restelli and Giraldo, 2009; Christopher J. Vogl *et al.*, 2019; Weller *et al.*, 2013; Zhang *et al.*, 2016a), which imposes strong time-step

restrictions and are not relevant for climate phenomena for example.

Within this scenario, SIMEX is a modification of the IMEX method. The advantage in this new class of methods is the absence of the need to iterate the implicit equation until a really low residue level. The residue that remains from the implicit equation is integrated along with the explicit part of the decomposition, thus keeping the method's precision. Due to the possibility of using less iterations, SIMEX has the potential to obtain precise solutions, as in IMEX, but faster (Rodrigues, 2017).

In this work SIMEX method is applied to Burgers' 1D equation as a preliminary test and the results are compared with the ones obtained from the classical 4^{th} order explicit Runge-Kutta. The spatial discretization is done by a compact finite difference method (Lele, 1992), as in the GATT's DNS. Solutions quality and method's formal order of convergence are verified by means of a manufactured solution. All the implementation and data analysis was done in Python. In the next step of this project SIMEX will be implemented in the GATT's DNS code which is written in FORTRAN 90.

2. METHODOLOGY

2.1 1D Burgers' Equation

Burgers' equation is an advection-diffusion PDE which is often used as a simplification for the Navier-Stokes equation, it is expressed as follows:

$$\frac{\partial u}{\partial t} + f(u)\frac{\partial u}{\partial x} = \gamma \frac{\partial^2 u}{\partial x^2} + \Psi(x,t) \tag{1}$$

where γ is a positive real constant, Ψ is a source term and $f(u)$ defines the advection speed, which here assumes two different profiles in order to get a linear or a non-linear equation. The former one is obtained with $f(u) = u$ and the last one with a constant profile $f(u) = c$.

2.2 Manufactured Solutions

In the present work the method of manufactured solutions is used for code verification. It consists of formulating an analytical solution by adding a source term to the original equation so that the numerical solution can be compared with the analytical one. For the one-dimensional PDE problem the chosen analytical function is given by:

$$U_{man}(x,t) = e^{\sin(\alpha x - at)}. \tag{2}$$

Where α and a represents the wave number and angular frequency respectively. This solution is chosen so that U_{man} is always positive, has infinite derivatives and oscillates in space and time.

The source term Ψ is easily calculated by

$$\Psi(x,t) = U_t + f(U)U_x - \gamma U_{xx}. \tag{3}$$

Where the subscript stands for the derivative with respect to that variable and the man subscript was suppressed to simplify notation.

2.3 Compact Finite Differences

Spatial derivatives are calculated with a spectral-like compact finite differences method described by (Lele, 1992), where centered stencil is defined as follows:

$$\beta(u'_{i+2} + u'_{i-2}) + \alpha(u'_{i+1} + u'_{i-1}) + u'_i = a\frac{u_{i+1} - u_{i-1}}{2h} + b\frac{u_{i+2} - u_{i-2}}{4h} + c\frac{u_{i+3} - u_{i-3}}{6h}, \tag{4}$$

where h is the mesh spacing. The correspondent modified wavenumber w' is

$$\omega'(\omega) = \frac{a\sin(\omega) + (b/2)\sin(2\omega) + (c/3)\sin(3\omega)}{1 + 2\alpha\cos(\omega) + 2\beta\cos(2\omega)}. \tag{5}$$

Close to the boundaries, the stencil is shifted and coefficients are adjusted accordingly, as an example, the first derivative at the boundary can be calculated from the following relation:

$$u'_1 + \alpha u'_2 = \frac{1}{h}(au_1 + bu_2 + cu_3 + du_4) \tag{6}$$

This system may be represented as

$$Au' = Bu, \tag{7}$$

where A and B are $N \times N$ square matrices with N being the number of grid points. The entries of B are proportional to h^{-1}.

Its worth mentioning that matrices A and B are narrow band-diagonal matrices. By choosing $\beta = c = 0$ in Eq. 4 it is possible to obtain up to a sixth-order tridiagonal scheme whose coefficients are determined by matching Taylor series. This work uses a fourth-order tridiagonal scheme with $w = 1.8$. When necessary, we will formally write

$$u' = A^{-1}Bu, \tag{8}$$

although Eq. 7 is used for computational purposes.

2.4 ESDIRK-IMEX

IMEX is a class of methods that combines an explicit with an implicit integrator and to do so it decomposes the right-hand-side of the ODE as the sum of two functions

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(t, \mathbf{y}) + \mathbf{g}(t, \mathbf{y}), \tag{9}$$

where f is the explicit part of the decomposition and g the implicit part. This allows stiff terms to be handled implicitly while non-stiff terms are placed in the explicit part. For IMEX methods, it is imperative to solve the implicit equation up to a given precision level in order to avoid introducing errors. This task is done by an iterative method that will iterate until de precision requirement is fulfilled or the maximum iterations number is reached, here we call this a Solver.

Explicit-first-stage singly diagonally implicit Runge-Kutta (ESDIRK) is a particular case of the DIRK class of implicit methods, for which an extensive review can be found in (Christopher A. Kennedy, 2016). The ESDIRK-IMEX implicit equation for the ith stage has the following form

$$\tilde{\mathbf{y}}_i - h\gamma\mathbf{g}(t_n + c_i h,\ \tilde{\mathbf{y}}_i) = \mathbf{y}_n + h\sum_{j=1}^{i-1}(a_{i,j}\mathbf{k}_j + \tilde{a}_{i,j}\tilde{\mathbf{k}}_j), \qquad i = 2, \ldots, \nu. \tag{10}$$

Where $\mathbf{y}_n$ is the current approximation for $\mathbf{y}(t_n)$, h is the step size and γ is the *tableau* diagonal's value. $\tilde{\mathbf{y}}_i$ is the current stage approximation, which is obtained by a Solver S within some predefined residue tolerance. Finally, the next step approximation $\mathbf{y}_{n+1}$ is computed as follows:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{j=1}^{\nu} b_j(\mathbf{k}_j + \tilde{\mathbf{k}}_j). \tag{11}$$

2.5 IMEX Decompositions

Having in mind that the advective term is the stiffness source to this problem, it should now be clear that this is the one to be placed in the implicit part of the decomposition. To avoid solving a non-linear system of equations, a linarisation of the advective term can be made. Here 4 decompositions are presented, 3 for the non-linear burgers' equation and 1 for the linear case.

Table 1: IMEX Decompositions for the linear and non-linear burgers' equation

	$g(u', u, x, t)$	$f(u'', u', u, x, t)$
Decomposition 1	$-u\frac{\partial u}{\partial x}$	$\gamma\frac{\partial^2 u}{\partial x^2} + \Psi(x,t)$
Decomposition 2	$-c\frac{\partial u}{\partial x}$	$-(u-c)\frac{\partial u}{\partial x} + \gamma\frac{\partial^2 u}{\partial x^2} + \Psi(x,t)$
Decomposition 3	$-u_0\frac{\partial u}{\partial x}$	$-(u-u_0)\frac{\partial u}{\partial x} + \gamma\frac{\partial^2 u}{\partial x^2} + \Psi(x,t)$
Decomposition 4	$-c\frac{\partial u}{\partial x}$	$\gamma\frac{\partial^2 u}{\partial x^2} + \Psi(x,t)$

Here the independent variables are the space x and temporal t dimensions and $u \equiv u(x,t)$ is the dependent variable. In this work, g and f are functions of u, its first and second derivatives and also x and t.

2.6 ESDIRK-Stepwise-IMEX

SIMEX (Rodrigues, 2017) is a new implicit-explicit integration method that can be seen as an improved IMEX, where a within-step decomposition adjustment is proposed with the intent of removing the need of getting precise solutions for the implicit equations. This new decomposition is named Residual Balanced Decomposition (RBD) and it is directly related to the traditional IMEX decomposition, Eq. 12. The job once done by a Solver is now done by a *implicit step filter*,

or *Filter* for short, which is defined as a predefined successive iterations of one or more iterative methods. For example, one can define a Filter as 2 Newton iterations. Thus, solvers and filters are distinguished because neither precision nor convergence are required from a filter.

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t,\mathbf{y}) + \mathbf{g}(t,\mathbf{y}) = \mathbf{f}^{rbd}(t,\mathbf{y}) + \mathbf{g}^{rbd}(t,\mathbf{y}). \tag{12}$$

The implicit equation that has to be solved at each SIMEX implicit step is exactly the same as the one for IMEX, Eq. 10. After that, the remaining residue has to be managed properly and that's what motivates the following definition for the RBD decomposition:

$$\mathbf{g}^{rbd}(t,\mathbf{y}) = \frac{\mathbf{y} - \mathbf{y}_n - \mathcal{F}^{-1}(\mathbf{y} - \mathbf{y}_n, t)}{h\gamma}. \tag{13}$$

Where $\mathcal{F}^{-1}$ is the inverse filter $\mathcal{F}$ and,

$$\mathbf{f}^{rbd}(t,\mathbf{y}) = \mathbf{f}(t,\mathbf{y}) + \mathbf{g}(t,\mathbf{y}) - \mathbf{g}^{rbd}(t,\mathbf{y}). \tag{14}$$

Now the question that arises is how to calculate the inverse filter, but even when possible it is certainly not a quick job. Altough its existence being important for the theoretical justification of the method, fortunately, there's a practical way to evaluate $\mathbf{g}^{rbd}$ and $\mathbf{f}^{rbd}$ without the need of computing $\mathcal{F}^{-1}$. So, let the value returned by the filter, Eq. 15, be the exact answer for the implicit equation, Eq. 10, when the function $\mathbf{g}$ is replaced by $\mathbf{g}^{rbd}$, Eq. 16. Mathematically,

$$\tilde{\mathbf{y}}_{i,rbd} = \mathcal{F}(\mathbf{rhs}, t) \tag{15}$$

such that

$$\tilde{\mathbf{y}}_{i,rbd} - h\gamma\mathbf{g}^{rbd}(t_n + c_i h,\ \tilde{\mathbf{y}}_{i,rbd}) = \mathbf{rhs}, \tag{16}$$

where $\mathbf{rhs} = \mathbf{y}_n + h\sum_{j=1}^{i-1}(a_{i,j}\mathbf{k}_j + \tilde{a}_{i,j}\tilde{\mathbf{k}}_j)$. Hence, solving for $\mathbf{g}^{rbd}$,

$$\mathbf{g}^{rbd}(t_n + c_i h,\ \tilde{\mathbf{y}}_{i,rbd}) = \frac{\tilde{\mathbf{y}}_{i,rbd} - \mathbf{rhs}}{h\gamma}. \tag{17}$$

And now $\mathbf{f}^{rbd}$ is completely determined by Eq. 14. After the calculations for the RBD decomposition the only thing missing is the variable update which is done exactly in the same way as in IMEX, Eq. 11.

The ESDIRK-SIMEX algorithm for a single step of size h starting from $t = t_n$ with step approximation $\mathbf{y}_n$ is presented bellow:

Algorithm 1 ESDIRK-SIMEX algorithm

$\mathbf{k}_1 = \mathbf{g}(t_n, \mathbf{y}_n)$
$\tilde{\mathbf{k}}_1 = \mathbf{f}(t_n, \mathbf{y}_n)$
for $i = 2;\quad i <= \nu;\quad i++$ **do**
  $\mathbf{rhs} = \mathbf{y}_n + h\sum_{j=1}^{i-1}(a_{i,j}\mathbf{k}_j + \tilde{a}_{i,j}\tilde{\mathbf{k}}_j)$
  $\tilde{\mathbf{y}}_{i,rbd} = \mathcal{F}$
  $\mathbf{k}_i = (\ \tilde{\mathbf{y}}_{i,rbd} - \mathbf{rhs})/h\gamma$
  $\tilde{\mathbf{k}}_i = \mathbf{f}(t_n + c_i h,\ \tilde{\mathbf{y}}_{i,rbd}) + \mathbf{g}(t_n + c_i h,\ \tilde{\mathbf{y}}_{i,rbd}) - \mathbf{k}_i$
end for
$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{j=1}^{\nu} b_j(\mathbf{k}_j + \tilde{\mathbf{k}}_j)$

An important but not so obvious observation that has to be made is about how to interpret this stepwise adjustment of the decomposition. Taking the filter's answer, Eq. 15, and putting it inside Eq. 10, it yields the following result:

$$\mathbf{g}(t_n + c_i h,\ \tilde{\mathbf{y}}_{i,rbd}) = \frac{\tilde{\mathbf{y}}_{i,rbd} - \mathbf{rhs} + \mathbf{res}}{h\gamma}, \tag{18}$$

Where $\mathbf{res}$ is the implicit equation residue. So,

$$\mathbf{g} - \mathbf{g}^{rbd} = \frac{\tilde{\mathbf{y}}_{i,rbd} - \mathbf{rhs} + \mathbf{res}}{h\gamma} - \frac{\tilde{\mathbf{y}}_{i,rbd} - \mathbf{rhs}}{h\gamma} = \mathbf{res}/h\gamma. \tag{19}$$

And Hence,

$$\mathbf{f}^{rbd} = \mathbf{f} + \mathbf{res}/h\gamma. \tag{20}$$

Therefore it is possible to say that the implicit equations residue is integrated along with the explicit part.

2.7 Iterative Methods for Implicit Equations

Iterative methods for implicit equations plays a big role in fully, or partially, implicit integration methods by affecting directly its computational efficiency. Thus, equating robustness and memory and processing costs is fundamental.

In this work we use Newton's step iteratios and to avoid the cost to compute and store the Jacobian matrix it is written in a way that there's no need of constructing the Jacobian matrix explicitly. This approach has the disadvantage of being customized for each case listed in section 2.5 The spatial dimension is represented as a n point uniform grid, so the variable u is written as a vector $\mathbf{u} = (u_1, \ldots, u_n)$. We introduce the notation $[u]$ to represent a diagonal matrix with $\mathbf{u}$ as the diagonal entries. The first derivative will be written as in Eq. 8.

Starting from the IMEX implicit equation, Eq. 10, written for $\mathbf{u}$, the problem is rewritten in the shape bellow.

$$\mathbf{F}(\mathbf{u}) = \mathbf{u} - \theta\mathbf{g}(\mathbf{u}) - \mathbf{rhs}, \tag{21}$$

where $\theta = h\gamma$ and $\mathbf{rhs} = h\sum_{j=1}^{i-1}(a_{i,j}\mathbf{k}_j + \tilde{a}_{i,j}\tilde{\mathbf{k}}_j)$. The Newton's step is obtained by the first order approximation of $F(\mathbf{u} + \mathbf{d})$, where $\mathbf{d}$ is small compared to $\mathbf{u}$. Eventually, one would end up with a linear system for the Newton's step $\mathbf{d}$, that can be solved with very efficient methods as Thomas algorithm since $\mathcal{A}$ is band matrix.

Table 2: Matrix Free Newton's Step

	$\mathcal{A}$	$\mathcal{B}$
Decomposition 1	$A[1/(\theta\mathbf{u})][1 + \theta\mathbf{u}_x] + B$	$A[1/(\theta\mathbf{u})](-F(\mathbf{u}))$
Decompositions 2 and 4	$A + \theta cB$	$-AF(\mathbf{u})$
Decomposition 3	$A[1/(\theta\mathbf{u}_0)] + B$	$-A[1/(\theta\mathbf{u}_0)]F(\mathbf{u})$

2.8 Butcher's Tableau for the used Runge-Kuttas

In this work some tests are performed with explicit and implicit-explicit Runge-Kutta methods. The pure explicit one is the classical 4^{th} order 4 stages ERK, also called as "RK4", which is used as reference of precision and execution time, the *tableau* is given in Eq. 22. Five ESDIRK-SIMEX pairs are used and they are listed bellow.

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array} \tag{22}$$

The pair "HCN" is a combination of Heun and Crank-Nicolson methods, which has 2 stages and a formal 2^{nd} order convergence rate. Also with 2^{nd} order, but 3 stages, there is the pair "ARS222" proposed by (Ascher *et al.*, 1997), using the choice of parameters $\gamma = 1 - \sqrt{2}/2$ and $\delta = 1 - 1/2\gamma$. The last three, which are proposed by Kennedy2001 under the names "ARK3(2)4L[2]SA-(ERK and ESDIRK)", "ARK4(3)6L[2]SA-(ERK and ESDIRK)" and "ARK5(4)8L[2]SA-(ERK and ESDIRK)", pairs are respectively: "ARK423", 3^{rd} order 3 stages; "ARK634", 4^{th} order 6 stages and "ARK845", 5^{th} order 8 stages.

3. PRELIMINARY RESULTS

3.1 Maximum Stable CFL

Some tables are presented in this section, each of them containing the maximum stable CFL number, CFL per stage and mean and maximum errors for each of the integration methods. Here the maximum CFL is defined as the highest CFL number that is possible to run the simulation without some kind of floating point error or with a resulting mean error bellow 10^{-2}. The uncertainty of these CFL numbers is smaller then 0.1.

The results are grouped in each table by the IMEX decomposition. The simulation parameters are: $n = 100$; $t = 6.5$ and $\alpha = \pi$. The last one is the manufactured solution wavenumber, the time is chosen to be long enough so the wave can travel at least one period across the spatial domain and the discretization is chosen in a way that it's not so refined but still far from coarse. There's a constant c, that when applicable, assumes the values 1.0, 0.5 and 6.0, in Tabs. 4, 5 and 7. Lastly, a variable time-steping is used in order to keep a constant CFL for the simulation.

These tests, Tabs. 3 to 7, aim identifying which is the highest stable CFL number for each combination of method and decomposition. The errors results are not suitable for a direct comparison, because even between methods with the same convergence order, the truncation errors are different.

Table 3: Classical 4^{th} order ERK and SIMEX with different tableaus. $tmax = 6.5$; $n = 100$; $\alpha = \pi$. Decomposition 1.

Method	Maximum CFL	CFL per Stage	Mean Error	Maximum Error
RK4	1.4	0.35	2.12597336391e-08	5.53578647366e-08
ARS222	1.4	0.47	2.12597336391e-08	5.31696036359e-05
ARKHCN	0.8	0.40	5.31696036359e-05	5.33712707123e-05
ARK423	1.2	0.30	1.35042794817e-06	3.2319289156e-06
ARK634	1.8	0.30	3.87861306506e-08	1.10751743665e-07
ARK845	2.2	0.27	2.20427199615e-08	5.64687081539e-08

Table 4: Classical 4^{th} order ERK and SIMEX with different tableaus. $tmax = 6.5$; $n = 100$; $\alpha = \pi$; $c = 0.5$. Decomposition 2.

Method	Maximum CFL	CFL per Stage	Mean Error	Maximum Error
RK4	1.4	0.35	2.12597336391e-08	5.53578647366e-08
ARS222	0.5	0.17	6.92703387128e-07	1.63334577019e-06
ARKHCN	0.5	0.25	7.44611892022e-07	1.71130735405e-06
ARK423	1.9	0.47	9.53206692589e-08	1.75987866058e-07
ARK634	1.7	0.28	2.86721002751e-08	6.62538695018e-08
ARK845	1.4	0.17	2.38761007099e-08	5.51128892523e-08

The obtained result for the RK4 was pretty consistent, showing a maximum CFL number of 1.4 and 1.3 for the non-linear and linear case respectively. All other tests were also consistent except for decomposition 2, where there was an increase of the maximum CFL number for ARK423 and a decrease for every other SIMEX *tableau*. It's not clear why this variations occurred, but the results were replicable even changing some parameters.

Table 5: Classical 4^{th} order ERK and SIMEX with different tableaus. $tmax = 6.5$; $n = 100$; $\alpha = \pi$; $c = 1.0$. Decomposition 2.

Method	Maximum CFL	CFL per Stage	Mean Error	Maximum Error
RK4	1.4	0.35	2.12597336391e-08	5.53578647366e-08
ARS222	0.6	0.2	2.16166959997e-06	5.23954220011e-06
ARKHCN	0.7	0.25	3.8248342577e-06	8.73097576637e-06
ARK423	1.5	0.37	2.92057404881e-07	1.42896969668e-06
ARK634	1.65	0.27	1.58342790316e-06	1.37241348535e-05
ARK845	1.75	0.22	2.38639030237e-08	5.27925102523e-08

Table 6: Classical 4^{th} order ERK and SIMEX with different tableaus. $tmax = 6.5$; $n = 100$; $\alpha = \pi$. Decomposition 3.

Method	Maximum CFL	CFL per Stage	Mean Error	Maximum Error
RK4	1.4	0.35	2.12597336391e-08	5.53578647366e-08
ARS222	1.5	0.5	2.25566960148e-05	6.23677746594e-05
ARKHCN	0.9	0.45	2.45096164958e-05	6.6238264234e-05
ARK423	1.2	0.30	9.51473536619e-07	2.13228285784e-06
ARK634	1.8	0.30	3.21259124247e-08	8.22027608205e-08
ARK845	2.2	0.27	2.07414324993e-08	5.24803230872e-08

The highest CFL result was obtained for the ARK845, 2.3 in Tab. 7, as expected, due to its higher order. The smaller value for each test is always obtained for the one with the lowest order and less stages, which was also expected. Disconsidering the outlier results for decomposition 2, Tabs. 4 and 5 , ARKHCN is always isolated with the worst CFL result. The pair ARS222 with only one more stage than ARKHCN was able to achieve up to 70%, even 100%, higher CFL numbers. Despite these differences in performance, both *tableau* pairs are strong candidates to beat the RK4's CPU time, due to its high maximum CFL numbers if taken in account the low number of stages. In the other hand, the pair ARK423 had a curious increase in stability at the decomposition 2 tests, what makes it a good candidate in this case.

Table 7: Classical 4^{th} order ERK and SIMEX with different tableaus. $tmax = 6.5$; $n = 100$; $\alpha = \pi$; $c = 6.0$. Decomposition 4.

Method	Maximum CFL	CFL per Stage	Mean Error	Maximum Error
RK4	1.3	0.32	6.68213096799e-08	8.38123477287e-08
ARS222	1.6	0.53	1.11847720401e-05	2.87018115995e-05
ARKHCN	0.8	0.40	1.04666337489e-05	2.37316281639e-05
ARK423	1.2	0.30	6.9711260396e-07	1.16764736607e-05
ARK634	1.8	0.30	1.59663717008e-07	8.44560643687e-06
ARK845	2.3	0.29	4.36773949997e-07	3.15404632989e-05

3.2 CPU Time

In this section the CPU time for the simulations are measured, with help of a Python's built in function, and then plotted against the mean error, both in logarithmic scale. For the first 4 figures, the simulations parameters are the same as in the previous section and than the instance with best result is tested more deeply. Each method runs at their maximum CFL and at two close by values. No parallelization technique was employed.

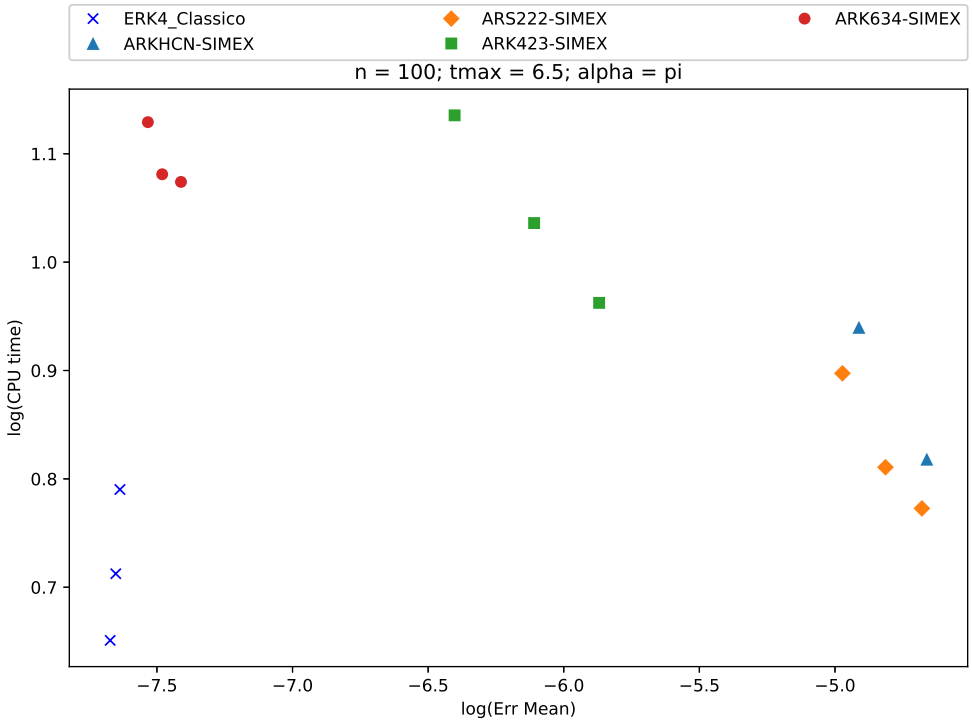

Figure 1: CPU time versus mean error for classical 4^{th} order ERK and SIMEX with different tableaus. $tmax = 6.5$; $n = 100$; $\alpha = \pi$. Decomposition 1

In a broad sense it can be observed that in 3 out of 5 instances, Fig. 1; 4 and 5, the 2^{nd} order *tableau* pairs presented the lowest execution time among the SIMEX. Furthermore the ARS222 consistently beats the ARKHCN. This shows that the cost of having one more stage is advantageous for ARS222. However, none of them was effectively faster than the RK4.

In the decomposition 2 cases, Figs. 2 and 3, the ARK423 was the quickest among the SIMEX, which confirms the expectation, but still a bit slower than the RK4. Another interesting thing that calls attention is that this 3^{rd} order SIMEX gives a solution almost as precise as the one given by the RK4. Lastly, one must call the attention to the 2^{nd} order SIMEX methods which got a technical tie with the RK4 on the last test, Fig. 5. However, it came with a cost of a mean error 3 orders of magnitude greater.

Motivated by the results shown in Fig. 5 where the two 2^{nd} order SIMEX basically equates the implicit steps' cost if compared to the reference explicit method, further investigation was made. Figure 5 shows that if the problem's stiffness is increased, SIMEX is more likely to be faster than the RK4. In this instance 3 different advection speeds are used and the simulation total time was raised to $tmax = 30$ in order to let the execution time scale grow. Both ARKHCN and

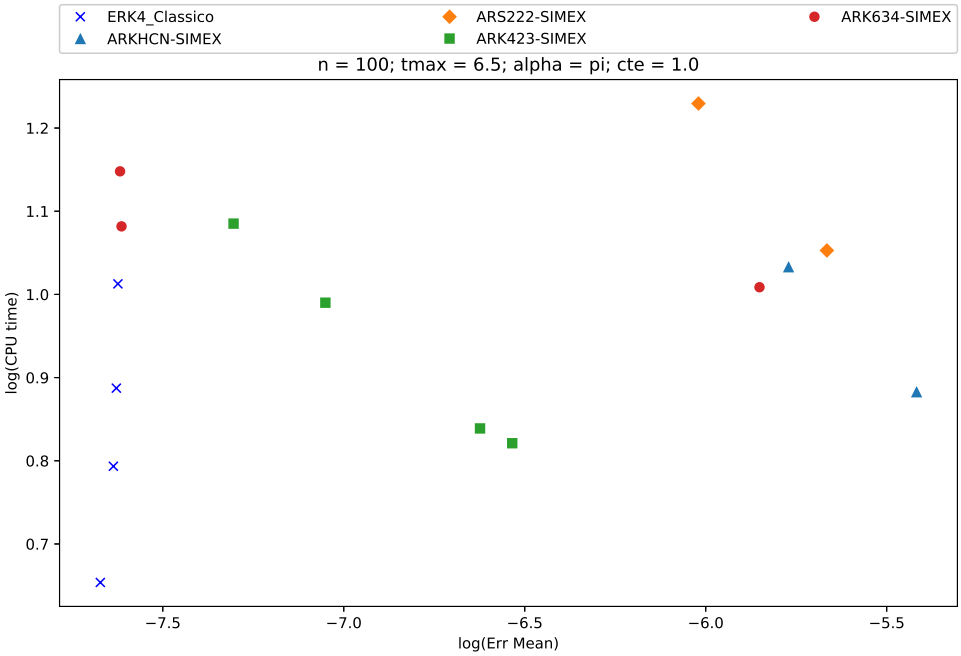

Figure 2: CPU time versus mean error for classical 4^{th} order ERK and SIMEX with different tableaus. $tmax = 6.5$; $n = 100$; $\alpha = \pi$; $c = 1.0$. Decomposition 2

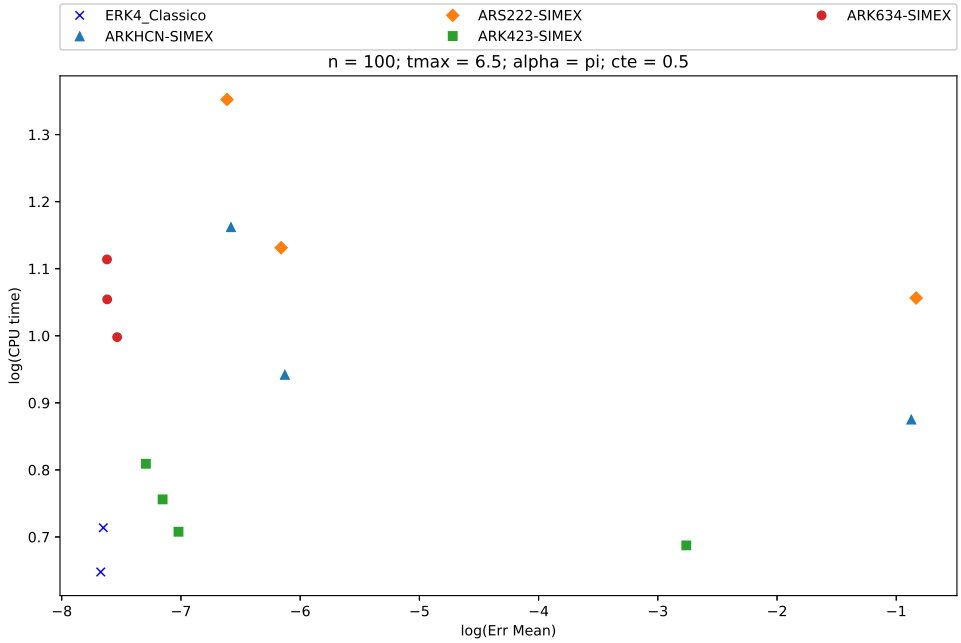

Figure 3: CPU time versus mean error for classical 4^{th} order ERK and SIMEX with different tableaus. $tmax = 6.5$; $n = 100$; $\alpha = \pi$; $c = 0.5$. Decomposition 2

ARS222 were faster than the reference method, but the second was consistently the fastest. In Fig. 6 ARS222 decreased the RK4's CPU time by 10.5%; in Fig. 7 by 13.8% and in Fig. 8 by 13.6%. The bigger stiffness does favor SIMEX efficiency, but there's no clear growth relation between them.

4. CONCLUSIONS

A test case successfully developed where the SIMEX method was implemented in a simplified version of the GATT's DNS code and than it was possible to measure the execution time and solution's quality and compare the results to the ones obtained with a classical 4^{th} order explicit Runge-Kutta. For such test the 1D Burgers' equation was used.

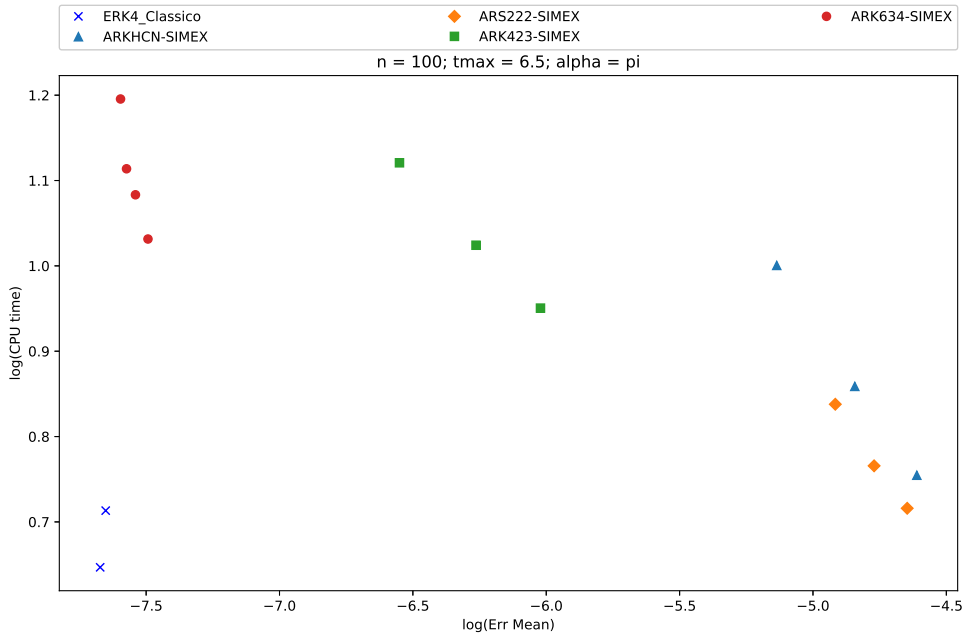

Figure 4: CPU time versus mean error for classical 4^{th} order ERK and SIMEX with different tableaus. $tmax = 6.5$; $n = 100$; $\alpha = \pi$. Decomposition 3

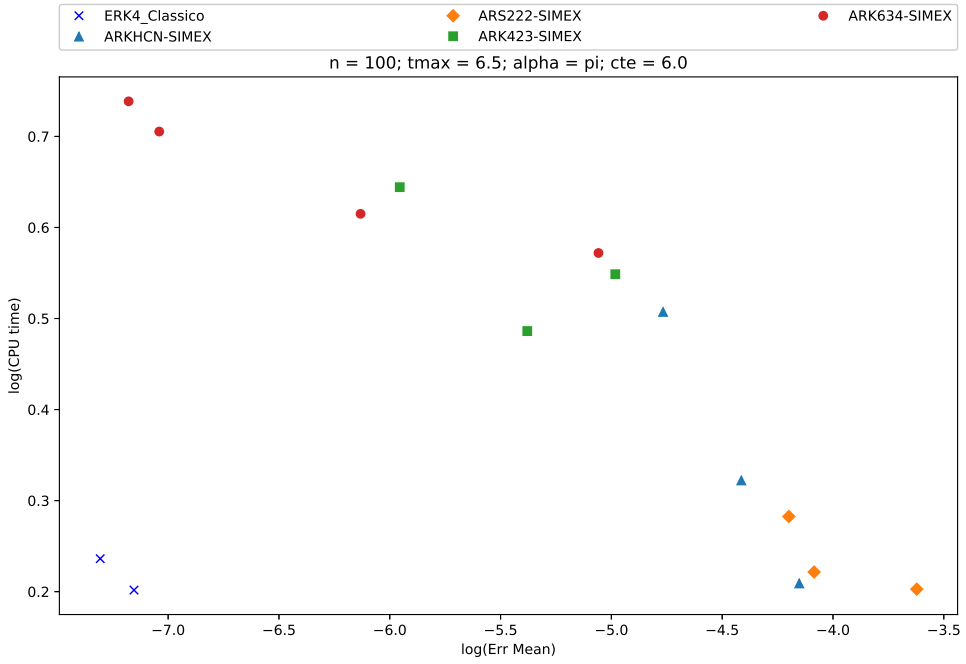

Figure 5: CPU time versus mean error for classical 4^{th} order ERK and SIMEX with different tableaus. $tmax = 6.5$; $n = 100$; $\alpha = \pi$; $c = 6.0$. Decomposition 4

Many *tableau* pairs were tested and they showed great impact on SIMEX efficiency but none of them was actually faster than the reference method for the non-linear case. The results for the linear Burgers' equation were favorable for SIMEX only when the convection speed was above a certain value, namely, the stiffness. Additionally, this positive results were only observable for the two second order *tableau* pairs. Despite the not so favorable results in these tests, it's not a bad sign for the SIMEX's performance with regard to future research, but the opposite. This conclusion is supported by a few things that need to be considered.

The model's stiffness is given by the convective term and when it is set in the implicit part the maximum step-size is now defined by the dissipative term; this doesn't happen in the DNS code. Considering the compressible Navier-Stokes equations, the terms related to convection and acoustic waves are the source of stiffness. At low Mach number flows, the

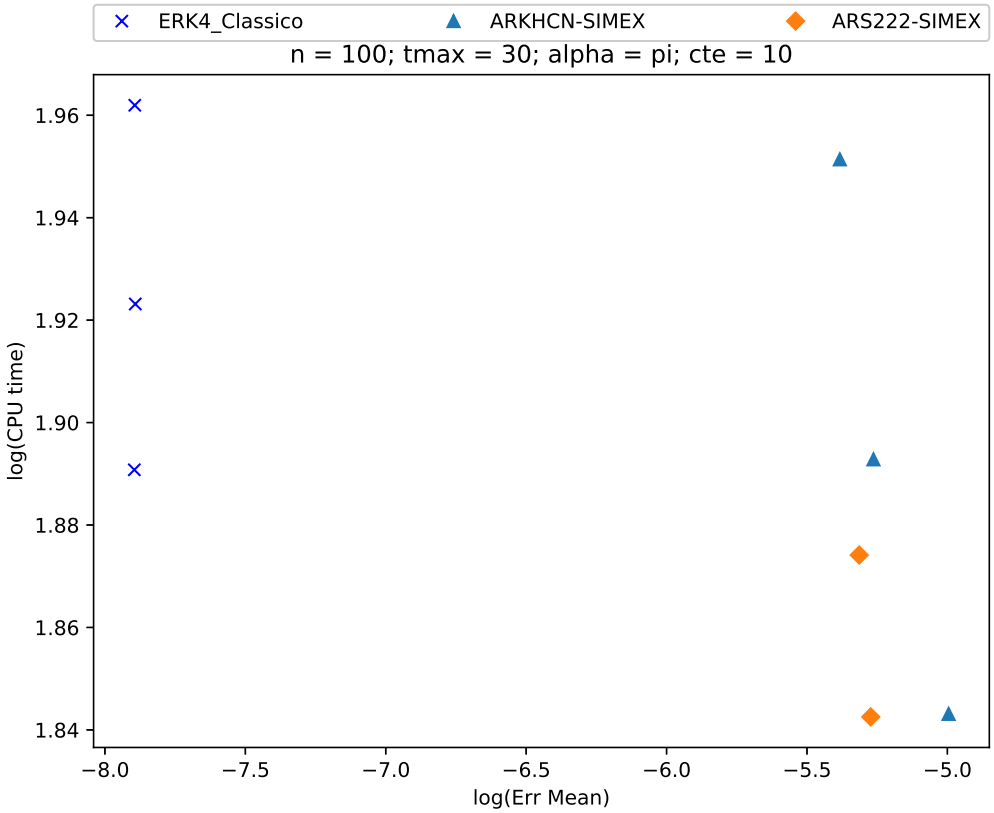

Figure 6: CPU time versus mean error for classical 4^{th} order ERK and SIMEX with different tableaus. $tmax = 30$; $n = 100$; $\alpha = \pi$; $c = 10.0$. Decomposition 4

propagation of acoustic waves imposes a much greater time-step restriction and a precise solution of this phenomena is not necessary unless one is studding aeroacoustics.

Besides, typical simulations use nonuniform meshes, where each spatial coordinate has a different refinement, which yields fairly different grid points density. Considering the standard 2D test case set in the DNS code, the grid points density is approximately one order of magnitude greater in the y axis. In other words, the grid stiffness is considerably higher in this direction. This intense refinement in the vertical direction in due to the necessity to resolve precisely the boundary layer and it's related phenomena.

Lastly, the choice of the decomposition is crucial for implicit-explicit method's efficiency, always seeking the best relation between higher time-stepping and implicit equation's complexity. The iterative method for implicit systems and the *tableau* pair also have huge impact on the efficiency. As a work guideline, SIMEX should be faster than the reference ERK no matter what *tableau* is used. But, in the other hand, the ones with less stages gave better results. That's a good indicative of what to investigate.

The next section presents a draft for the work's next phase containing the DNS governing equations, proposed IMEX decompositions and iterative methods.

5. NEXT STEPS

By the time the preliminary tests are ended, the next step is to implement the SIMEX method into the DNS code. A detailed explanation for the DNS code's structure is found in Mathias (2017).

The fluid's thermodynamic state is completely defined by the density (ρ), internal energy (e) and the three velocity components (u,v,w). Each variable depends on location (x,y,z) and time (t). The equations for the conservation of linear momentum, mass and energy are written in the non-conservative form, which solves for each individual variable mentioned above, simplifying the boundary conditions. The pressure p is calculated assuming an ideal gas.

The first proposed IMEX implicit part is to be the source terms of acoustic wave propagation in the y direction, due to

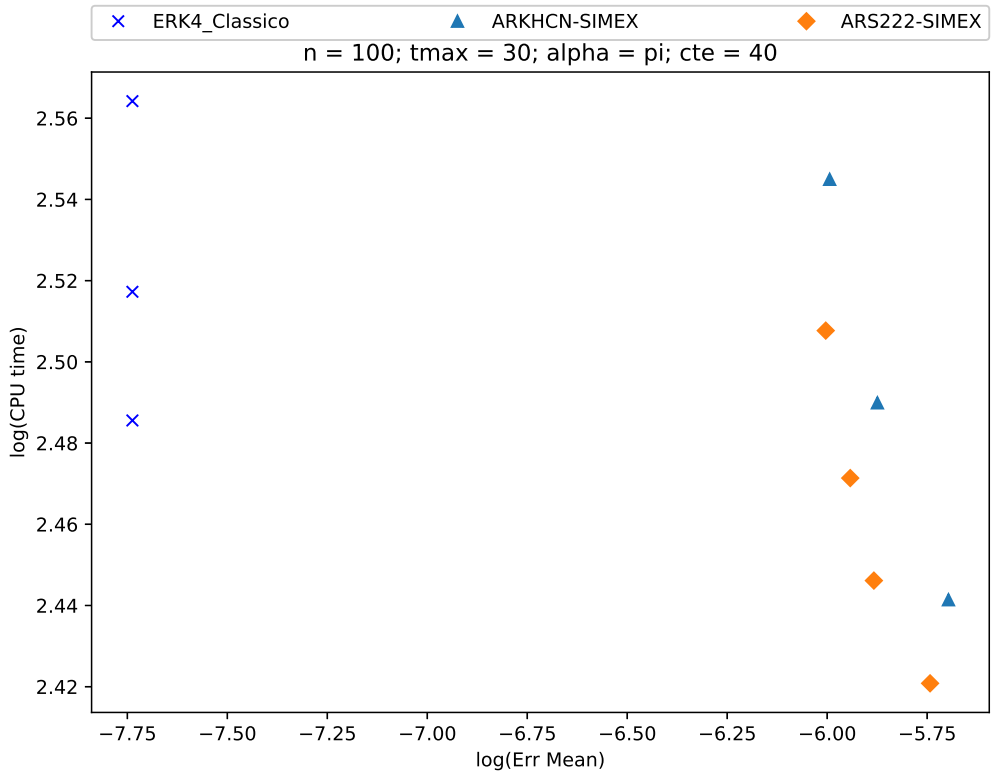

Figure 7: CPU time versus mean error for classical 4^{th} order ERK and SIMEX with different tableaus. $tmax = 30$; $n = 100$; $\alpha = \pi$; $c = 40.0$. Decomposition 4

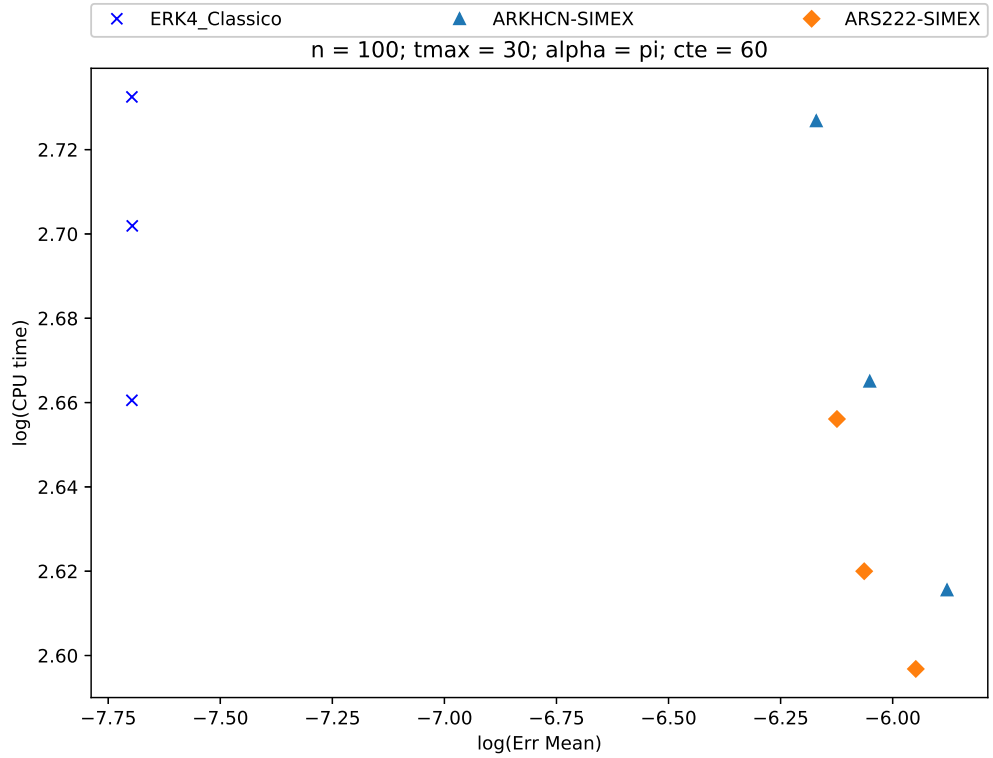

Figure 8: CPU time versus mean error for classical 4^{th} order ERK and SIMEX with different tableaus. $tmax = 30$; $n = 100$; $\alpha = \pi$; $c = 60.0$. Decomposition 4

refined grid in this direction. Hence, the implicit-explicit decomposition is defined as

$$g(\mathbf{y}) = \begin{pmatrix} -\rho_0 v_y \\ -\frac{\gamma-1}{\rho_0}(\rho_0 e_y + \rho_y e_0) \\ -\frac{p_0}{\rho_0} v_y \end{pmatrix} \quad \text{and} \quad f(\mathbf{y}) = \frac{d\mathbf{y}}{dt} - g(\mathbf{y}), \tag{23}$$

where the zero subscript as in ρ_0 represents the variable's value at the beginning of the integration step, which is kept constant through the entire step. The subscript y as in v_y denotes the derivative with respect to y, the constant γ, in the DNS context, is the thermodynamic relation c_p/c_v and

$$\mathbf{y} = \begin{pmatrix} \rho(y,t) \\ v(y,t) \\ e(y,t) \end{pmatrix}. \tag{24}$$

One good evidence that this proposed decomposition does group all the acoustic wave's source term that propagates in the y direction is to be able to retrieve the wave equation starting from these terms, which is true, indeed.

The last item to be considered in this initial approach is the choice of an iterative method for the implicit system. Could be of great use a method that is simple to code and has a low processing cost as Gauss-Seidel and Gauss-Jacobi, so it'll be possible to have the first insights about the SIMEX's efficiency in the DNS. These iterations requires little computational effort and hopefully the increase in the time-step will more than compensate that cost. As the time-step becomes larger and larger, the off diagonal terms become larger, which makes the system more difficult to solve. Hence it may be necessary to use more powerful methods.

6. ACKNOWLEDGEMENTS

G.B.F.B. received funding from Coordination for the Improvement of Higher Education Personnel (CAPES/Brazil). M.A.F.M. is sponsored by CNPq/Brazil, grant no. 307956/2019-9

7. REFERENCES

Ascher, U.M., Ruuth, S.J. and Spiteri, R.J., 1997. "Implicit-explicit runge-kutta methods for time-dependent partial differential equations". *Applied Numerical Mathematics*, Vol. 25, No. 2, pp. 151 – 167. ISSN 0168-9274. doi: https://doi.org/10.1016/S0168-9274(97)00056-1. Special Issue on Time Integration.

Bergamo, L.F., Gennaro, E.M., Theofilis, V. and Medeiros, M.A., 2015. "Compressible modes in a square lid-driven cavity". *Aerospace Science and Technology*, Vol. 44, pp. 125–134. ISSN 12709638. doi:10.1016/j.ast.2015.03.010. URL https://linkinghub.elsevier.com/retrieve/pii/S1270963815001030.

Bergamo, L.F., 2014. *Instabilidade hidrodinâmica linear do escoamento compressível em uma cavidade*. Master thesis, Universidade de São Paulo, São Carlos. doi:10.11606/D.18.2014.tde-28052014-164324. URL http://www.teses.usp.br/teses/disponiveis/18/18148/tde-28052014-164324/.

Bispen, G., Lukáčová-Medvid'ová, M. and Yelash, L., 2017. "Asymptotic preserving imex finite volume schemes for low mach number euler equations with gravitation". *Journal of Computational Physics*, Vol. 335, pp. 222 – 248. ISSN 0021-9991. doi:https://doi.org/10.1016/j.jcp.2017.01.020.

Christopher A. Kennedy, M.H.C., 2001. "Additive Runge-Kutta Schemes for Convection-Diffusion-Reaction Equations". Technical report, National Aeronautics And Space Administration, Virginia.

Christopher A. Kennedy, M.H.C., 2016. "Diagonally Implicit Runge-Kutta Methods for Ordinary Differential Equations. A Review". *National Aeronautics And Space Administration*. URL https://ntrs.nasa.gov/search.jsp?R=20160005923 2020-03-22T21:22:55+00:00Z.

Christopher A. Kennedy, M.H.C., 2018. "Higher-order additive Runge–Kutta schemes for ordinary differential equations". *Applied Numerical Mathematics*, p. 183–205. URL https://doi.org/10.1016/j.apnum.2018.10.007.

Christopher J. Vogl, A.S., Reynolds, D.R., Ullrich, P.A. and Woodward, C.S., 2019. "Evaluation of Implicit-Explicit Additive Runge-Kutta Integrators for the HOMME-NH Dynamical Core". *Journal of Advances in Modeling Earth Systems (JAMES)*.

Colavolpe, C., Voitus, F. and Bénard, P., 2017. "Rk-imex hevi schemes for fully compressible atmospheric models with advection: analyses and numerical testing". *Quarterly Journal of the Royal Meteorological Society*, Vol. 143, No. 704, pp. 1336–1350. doi:10.1002/qj.3008.

Cooper, G. and Sayfy, A., 1980. "Additive methods for the numerical solution of ordinary differential equations". *Mathematics of Computation*, Vol. 35, No. 152, pp. 1159–1172.

Crouzeix, M., 1980. "Une méthode multipas implicite-explicite pour l'approximation des équations d'évolution paraboliques". *Numerische Mathematik*, Vol. 35, No. 3, pp. 257–276.

David J. Gardner, J.E.G., Hamon, F.P., Reynolds, D.R., Ullrich, P.A. and Woodward, C.S., 2018. "Implicit–explicit (IMEX) Runge–Kutta methods for non-hydrostatic atmospheric models". *Copernicus Publications on behalf of the European Geosciences Union*.

Durran, D.R. and Blossey, P.N., 2012. "Implicit–explicit multistep methods for fast-wave–slow-wave problems". *Monthly Weather Review*, Vol. 140, No. 4, pp. 1307–1325.

Ghosh, D. and Constantinescu, E.M., 2016. "Semi-implicit time integration of atmospheric flows with characteristic-based flux partitioning". *SIAM Journal on Scientific Computing*, Vol. 38, No. 3, pp. A1848–A1875. doi: 10.1137/15M1044369.

Hofer, E., 1976. "A partially implicit method for large stiff systems of odes with only few equations introducing small time-constants". *SIAM Journal on Numerical Analysis*, Vol. 13, No. 5, pp. 645–663. doi:10.1137/0713054.

Kennedy, C.A. and Carpenter, M.H., 2003. "Additive runge–kutta schemes for convection–diffusion–reaction equations". *Applied Numerical Mathematics*, Vol. 44, No. 1, pp. 139 – 181. ISSN 0168-9274. doi:https://doi.org/10.1016/S0168-9274(02)00138-1.

Lele, S.K., 1992. "Compact finite difference schemes with spectral-like resolution". *Journal of Computational Physics*, Vol. 103, No. 1, pp. 16–42. ISSN 00219991. doi:10.1016/0021-9991(92)90324-R.

Martinez, A., 2016. *Towards natural transition in compressible boundary layers*. Phd. thesis, University of São Paulo.

Martinez, A. and Medeiros, M.F., 2016. "Direct numerical simulation of a wavepacket in a boundary layer at Mach 0.9". In *46th AIAA Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, Reston, Virginia, Vol. 414, pp. 1–33. ISBN 978-1-62410-436-7. ISSN 00221120. doi:10.2514/6.2016-3195. URL `http://arc.aiaa.org/doi/10.2514/6.2016-3195`.

Mathias, M.S., 2017. *Instability analysis of compressible flows over open cavities by a Jacobian-free numerical method*. Master thesis, University of São Paulo.

Restelli, M. and Giraldo, F.X., 2009. "A conservative discontinuous galerkin semi-implicit formulation for the navier–stokes equations in nonhydrostatic mesoscale modeling". *SIAM Journal on Scientific Computing*, Vol. 31, No. 3, pp. 2231–2257.

Rodrigues, S.B., 2017. "A shortcut for IMEX methods: integrate the residual explicitly". *eprint arXiv*, Vol. 1705.04870. URL `http://arxiv.org/abs/1705.04870`.

Silva, H.G., Souza, L.F. and Medeiros, M.A.F., 2010. "Verification of a mixed high-order accurate dns code for laminar turbulent transition by the method of manufactured solutions". *International Journal for Numerical Methods in Fluids*, Vol. 64, No. 3, pp. 336–354. doi:10.1002/fld.2156. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.2156`.

Wang, H., Shu, C.W. and Zhang, Q., 2015. "Stability and error estimates of local discontinuous galerkin methods with implicit-explicit time-marching for advection-diffusion problems". *SIAM Journal on Numerical Analysis*, Vol. 53, No. 1, pp. 206–227. doi:10.1137/140956750.

Weller, H., Lock, S.J. and Wood, N., 2013. "Runge–kutta imex schemes for the horizontally explicit/vertically implicit (hevi) solution of wave equations". *Journal of Computational Physics*, Vol. 252, pp. 365 – 381. ISSN 0021-9991. doi:https://doi.org/10.1016/j.jcp.2013.06.025.

Zhang, H., Sandu, A. and Blaise, S., 2016a. "High order implicit-explicit general linear methods with optimized stability regions". *SIAM Journal on Scientific Computing*, Vol. 38, No. 3, pp. A1430–A1453.

Zhang, H., Sandu, A. and Blaise, S., 2016b. "High order implicit-explicit general linear methods with optimized stability regions". *SIAM Journal on Scientific Computing*, Vol. 38, No. 3, pp. A1430–A1453. doi:10.1137/15M1018897.

8. RESPONSIBILITY NOTICE

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