Turbulence, Heat and Mass Transfer 10

Cost-vs-accuracy analysis of self-adaptive time-integration methods

J. Plana-Riu, F.X. Trias, C.D. Pérez-Segarra and A.Oliva
Heat and Mass Transfer Technological Centre, Technical University of Catalonia, ESEIAAT, Carrer de Colom 11, 08222 Terrassa (Barcelona), Spain, josep.plana.riu@upc.edu

Abstract – The integration in time of the semi-discrete Navier-Stokes equations has been historically bounded by the classic CFL condition, and later on, an innovative method has been defined by ensuring the eigenvalues of the method lay in the boundary of the stability region, yet the effects of this in the resolution have not been tested. In this abstract, an efficiency region in a set of schemes is defined so that the method can alternate among methods in order to keep the maximum efficiency subject to a set of conditions, such as a minimum order of accuracy or a minimum numerical dissipation.

1. Introduction

Solving and understanding turbulence has been one of the problems in which most efforts have been applied within the fluid dynamics community, which is mathematically modelled by the Navier-Stokes equations. Within the framework of the finite volume method (FVM), its space-discretized form is the starting point for most of the methods. This form reads, using the notation from [1], as follows,

$$M\mathbf{u}_s = \mathbf{0}_c, \tag{1}$$

$$\Omega\frac{d\mathbf{u}_c}{dt} + C(\mathbf{u}_s)\mathbf{u}_c - D\mathbf{u}_c + \Omega G_c\mathbf{p}_c = \mathbf{0}_c, \tag{2}$$

where M is the face-to-cell divergence operator, Ω_c is a diagonal matrix containing the cell volumes so that $\Omega = I_3 \otimes \Omega_c$, C_c is the cell-to-cell convective operator so that $C = I_3 \otimes C_c$, D_c is the cell-to-cell diffusive operator so that $D = I_3 \otimes D_c$, G_c is the cell-to-cell gradient operator, $\mathbf{u}_s$ is the velocity field defined at the faces, and I_3 is the identity matrix of size 3. Traditionally, the solution of the semi-discretized equations has been obtained with a projection method [2], in which multiple schemes can be applied to advance the equations in time (e.g. second order Adams-Bashforth ,AB2; second- and third- order Runge-Kutta schemes, RK2,RK3; etc.), with their variations in their properties.

It is well-known that by performing the numerical solution of the Navier-Stokes equations, it is possible to introduce artificial dissipation due to the discretizations used. In the work of Verstappen and Veldman [3], and Trias et al. [1], the numerical dissipation due to the space discretization is treated, leading to symmetry-preserving space schemes both in staggered and collocated methods. Nonetheless, there is a reduced set of publications in which the numerical dissipation due to the time discretization is treated. Sanderse [4] proposed the use of symplectic (implicit) Runge-Kutta schemes in order to completely preserve energy. In order not to deal with implicit time integration, Capuano et al. [5] proposed pseudosymplectic schemes, which at the same time are explicit, making them more efficient. Both of these possible sources of numerical dissipation can be put altogehter in a so-called kinetic energy budget, in which the weight of this dissipation can be compared against the physical dissipation.

In order to properly set the time-step, Trias and Lehmkuhl [6] first proposed making use of the stability region so that the integration procedure can be optimized, having the eigenvalues of the method, bounded with Gershgorin's theorem, to be in the boundaries of its stability region. This led to bigger time-steps while still having stable simulations when compared to the classical CFL method.

2. Runge-Kutta schemes applied to the incompressible Navier-Stokes equations

Starting from the semi-discrete Navier-Stokes equations, Sanderse and Koren [7] proposed the following method for the integration with Runge-Kutta (RK). By applying the incompressibility condition to the momentum equation, and defining the function $F(\mathbf{u}_s)\mathbf{u}_c = \Omega^{-1}(D\mathbf{u}_c - C(\mathbf{u}_s)\mathbf{u}_c)$ and rearranging the equation, it can be found that for a domain discretized in n cells

$$\frac{d\mathbf{u}_c}{dt} = (I_n - GL^{-1}M)F(\mathbf{u}_s)\mathbf{u}_c, \tag{3}$$

where $I_n - GL^{-1}M$ is the so-called projection operator P, which leads to

$$\frac{d\mathbf{u}_c}{dt} = PF(\mathbf{u}_s)\mathbf{u}_c. \tag{4}$$

In this system of ordinary differential equations, the reconstruction of the projection operator would imply a large computational time as the product of two sparse matrices is costly. On the other hand, a projection method such as the one proposed by Chorin [2] can be applied without the requirement of computing this projection operator, making the implementation more efficient and simple.

Summarizing, an s-stage explicit RK can be applied to the integration of the Navier-Stokes equations as

$$\mathbf{u}_i^* = \mathbf{u}_n + \Delta t \sum_{j=1}^{i-1} a_{ij}\mathbf{F}_j, \;\; L\Psi_i = \frac{1}{\Delta t} D\mathbf{u}_i^*, \;\; \mathbf{u}_i = \mathbf{u}_i^* - \Delta t G\Psi_i, \quad i = 1,\ldots,s \tag{5}$$

$$\mathbf{u}_{n+1}^* = \mathbf{u}_n + \Delta t \sum_{i=1}^{s} b_i\mathbf{F}_i, \;\; L\Psi_{n+1} = \frac{1}{\Delta t} D\mathbf{u}_{n+1}^*, \;\; \mathbf{u}_{n+1} = \mathbf{u}_{n+1}^* - \Delta t G\Psi_i, \tag{6}$$

where $\mathbf{F}_i = \Omega^{-1}(D - C(\mathbf{u}_{s,i}))\mathbf{u}_{c,i}$, a_{ij}, b_i correspond to the Butcher tableau of the used method for an explicit Runge-Kutta method, $\mathbf{u}_i^*$ corresponds to the predictor velocity, and Φ to a pressure-like variable which corresponds to a first order approximation to the pressure.

The initial velocity field will usually be incompressible from an analytical point of view, yet in order to have incompressible initial conditions, this field should be arranged so that $M\mathbf{u}_{s,0} = \mathbf{0}_c$ is fulfilled. In order to do so, the discretized initial condition will be treated as a predictor velocity in Eq. (6), computing an initial pressure field as well as a projected (and now incompressible) velocity field.

If the coefficients from the Butcher tableau satisfy a certain conditions, a pseudosymplectic scheme can be obtained [5] in which energy is conserved for an order higher than the order of accuracy (e.g. a 3p5q(4) method corresponds to a third-order in accuracy, fifth in energy conservation, by using four stages).

Note that these methods will not have either the stability region $R(z), z \in \mathbb{C}$ of a classical RK3 or RK4, which can be easily computed by (7). Instead, their stability regions will need to be computed with its definition (8), in which $b = (b_1\ b_2\ \dots\ b_s)^T$, $\mathbf{1} = (1\ 1\ \dots\ 1)^T \in \mathbb{R}^s$, and $A = [a_{ij}] \in \mathbb{R}^{s\times s}$.

$$R(z) = 1 + \sum_{p=1}^{s} \frac{1}{p!} z^p, \tag{7}$$

$$R(z) = 1 + zb^T (I_s - zA)^{-1}\mathbf{1}. \tag{8}$$

As an example, according to Capuano et al. [5], the 3p5q(4) pseudosymplectic scheme is parameter-dependant, the domain on which is discussed in the paper. Hence, the envelope of the stability regions for all possible parameters will determine the stability region of the so-called α-3p5q(4) method, and can be compared against the RK3 and RK4 schemes, as in Figure 1.

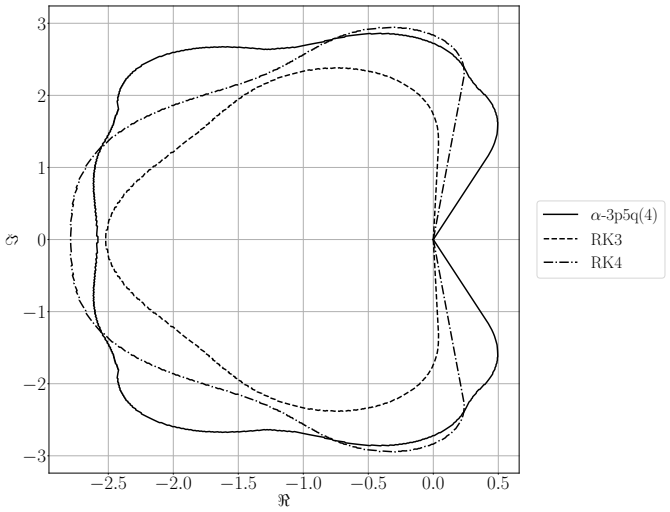

Figure 1: Comparison of the stability regions for a RK3, RK4 and the parameter-dependant α-3p5q(4).

Once the stability region of the method is computed, the eigenbounds of the operator $F(\mathbf{u}_s)$ need to be computed. Assuming a symmetry-preserving space discretization, the convective operator will translate to a skew-symmetric matrix in the discrete domain, whereas the diffusive operator will become a symmetric matrix. This means that the eigenbounds of the former, λ_C, will lay in the imaginary axis within the complex plane, while the eigenbounds of the latter, λ_D, will lay in the real axis. Moreover, being the discrete operator semi-negative definite, these eigenbounds will correspond to $\mathbb{R}^-$. Hence, the eigenbounds of the operator F will correspond to

$$\lambda_F \leq -|\lambda_D| + i\lambda_C, \tag{9}$$

and thus these λ_D and λ_C can be computed independently. In order to do so, the Gershgorin circle theorem as done by Trias and Lehmkuhl [6] will be applied so that Δt_s is obtained. Nonetheless, the actual time-step that will be used for the simulations will correspond to

$$\Delta t = f_{\Delta t} \Delta t_s, \tag{10}$$

where $f_{\Delta t} \leq 1.0$ will act as a Courant-like number for the self-adaptive time-step length, since it will affect in other parameters of the simulation such as its energy budgets, for instance.

3. From stability regions to computational efficiency regions

The concept of stability regions is widely spread in the numerical solution of ODEs, being Figure 1 a clear example of the concept, as it bounds the maximum value the eigenvalues can take so that the integration remains stable. Usually, the time-integration scheme does not change during the run. Nonetheless, the efficiency region is based on using a closed set of families of schemes (e.g. RK2, RK3, RK4, α-3p5q) so that the most efficient scheme (under certain conditions explained later on) is used at every time. This could be set by normalizing the stability regions, which radius is directly proportional to the maximum stable time-step, by the time required for the computation of a time-step. Hence, this time-step computation time can be directly estimated with the generalized implementation in the in-house code framework,

$$T_{\Delta t}(s) = sT_{SLAE} + 33sT_{SpMV} + s\left(24 + 3\frac{s-1}{2}\right)T_{axpy} + 10sT_{axty}, \tag{11}$$

which will work for an arbitrary number of stages s, where T_{SLAE} corresponds to the time spent computing the system of linear algebraic equations (SLAE), which corresponds to a Poisson equation; T_{axpy} is the time spent for a linear combination of vectors, and T_{SpMV} corresponds to the time spent for a sparse matrix-vector product. Hence, the radius of the efficiency region would be proportional to the maximum $\dfrac{\Delta t}{T_{\Delta t}^{\text{Method}}}$. Nonetheless, in order to simplify the notation,

$$T_{\Delta t}^{\text{Method}} = \tau_{\text{Method}} T_{\Delta t}^{\text{Euler}} = \tau_{\text{Method}} T_{\Delta t}(1) = \tau_{\text{Method}}(T_{SLAE} + 27T_{axpy} + 33T_{SpMV} + 10T_{axty}), \tag{12}$$

where τ_{Method} is the ratio between $T_{\Delta t}^{\text{Method}}$ and $T_{\Delta t}(1)$, so that the efficiency region will be proportional to $\dfrac{\Delta t}{\tau_{\text{Method}}}$. Note that (11) only depends on the number of stages, and not on the coefficients, and thus τ_{Method} will depend only in s as well. Hence, the efficiency region for a set of standard Runge-Kutta schemes (Euler, RK2, RK3, RK4), with the stability region computed with (7) has been obtained in Figure 2. Note that the maximum efficiency for low angles within the complex plane is obtained with an Euler scheme, and thus, when adding the condition of minimum second-order in time integration the efficiency for low ϕ is notably reduced, while keeping the same efficiency at the most convective cases, and setting a limiting condition in the efficiency region.

Hence, the general method will englobe a set of schemes among which it will be capable of chosing one of the methods from the set such that the efficiency region is maximized in that case, being subject to all of the possible constraints defined by the method.

Note that the method can be straightforwardly extended to multi-step schemes such as an AB2 or AB3, which would have similar computation times compared to an Euler scheme, given that there will not be any additional stages compared to the presented Runge-Kutta methods.

The presented method will slightly modify the computation of the time-step compared to

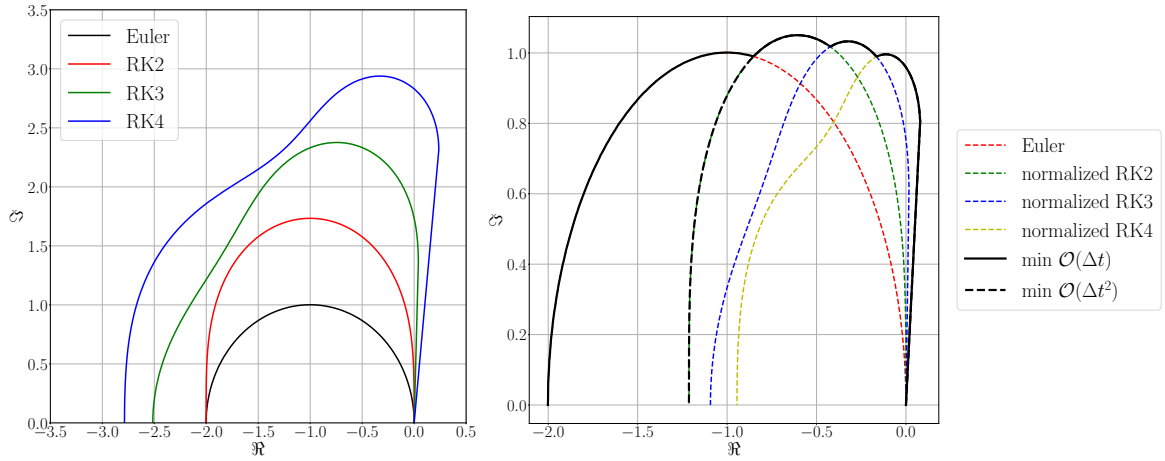

Figure 2: Stability (left) and efficiency (right) regions for a set of standard RK schemes. The efficiency region is computed considering that the solution of the Poisson equation corresponds to an 80% of the iteration time for an Euler scheme.

the classical computation of a self-adaptive scheme. In order to do so, a *pool* of Runge-Kutta schemes will be selected so that they are candidates to integrate the following time step.

Algorithm 1: Self-adaptive efficiency time-step scheme

Data: *Pool* of schemes, $\{\text{Scheme}_1,\text{Scheme}_2,\ldots,\text{Scheme}_n\}$,Eigenbounds $\{\lambda_C,\lambda_D\}$, Scale factor, $f_{\Delta t}$
Result: Time-step, Δt

$i := 1$;
$i_s := i$; $\Delta t_s := 0.0$;
while $i \leq n$ **do**
  $\Delta t_{s,i} = \Delta t_s(\text{Scheme}_i, \lambda_C, \lambda_D)$;
  if $\Delta t_{s,i}/\tau_{Scheme_i} \geq \Delta t_s$ **then**
    $\Delta t_s = \Delta t_{s,i}/\tau_{\text{Scheme}_i}$;
    $i_s := i$;
return $\Delta t := f_{\Delta t}\Delta t_s(Scheme_{i_s}, \lambda_C, \lambda_D)$;

At the beginning of every time-step, as per Alg. 1 the scheme will calculate the eigenbounds, λ_C and λ_D, as usual. Afterwards, the stability time-step length for all of them will be computed and scaled with the τ_{Method} and the largest scaled value will be selected so that its corresponding scheme will be the selected candidate to integrate that time-step, named i_s, and thus the time-step will be set by applying the classical Gershgorin with the stability region obtained for the i_s-th scheme.

4. Numerical experiments

The numerical experiments carried out to validate the method will correspond to a three-dimensional Taylor-Green vortex (TGV) problem. In this case, the velocity field used to initialize the field

corresponds to

$$u_{x,0} = U_0 \frac{2}{\sqrt{3}} \sin(x)\cos(y)\cos(z) \tag{13a}$$

$$u_{y,0} = U_0 \frac{2}{\sqrt{3}} \cos(x)\sin(y)\cos(z) \tag{13b}$$

$$u_{z,0} = U_= \frac{2}{\sqrt{3}} \cos(x)\cos(y)\sin(z) \tag{13c}$$

in a cubic domain with side 2π and will be run until $t = 20\frac{2\pi}{U_0}$, following the test cases from Capuano, Coppola and Luca [8]. This test will be run for the schemes listed in Tab. 1, with their properties, as well as for an efficient set up with a pool of schemes corresponding to all of the schemes tested individually.

Table 1: List of the Runge-Kutta schemes used.

Method	Num. stages, s	Ord. accuracy, p
Euler	1	1
Heun RK2	2	2
Heun RK3	3	3
Standard RK4	4	4
4p7q(6) [8]	6	4

The case was tested for a $\mathrm{Re} = 3000$ and for $f_{\Delta t} = 0.1$. First of all, in the run with the efficient set up, only the methods with $s \leq 4$ were actually used, so the 4p7q(6) scheme was not used at any time in the simulation (Fig. 3), even though it provides the largest possible time-step.

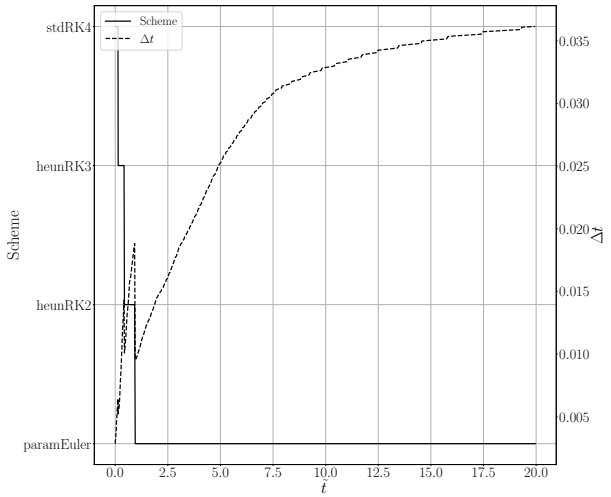

Figure 3: Evolution of the time-step as well as the schemes used in the numerical solution of a $\mathrm{Re} = 3000$ TGV with $f_{\Delta t} = 0.1$.

This can be easily explained by the fact that even though it has 6 stages, and 4th order in accuracy, which leads to a bigger stability region than its 4th order in accuracy counterpart standard RK4, it is not big enough for a $\tau_{4p7q(6)} \approx 6.0$ and thus is not used in the whole simulation.

Moreover, the shift towards a more diffusive problem in the whole duration of the simulation, since the problem is left to decay for the whole run, will lead to the use of schemes with a lower number of stages (and lower accuracy) since ϕ will lower with the evolution of time, which is observed in Fig. 3. Nonetheless, note that no limitations on the accuracy of the scheme were imposed and thus, as seen in Fig. 2, for most of the low ϕ range, the most efficient scheme will be the Euler, and thus it will be the most used as shown in Fig. 3.

With regards to the wall clock time savings due to the implementation of this method compared to the classical methods computed using with exactly the same conditions $f_{\Delta t} = 0.1$ and $\mathrm{Re} = 3000$, with 4 CPU cores, it takes 498.70 s, which implies an improvement of a 47% against a whole simulation with an Euler scheme (951.78 s), a 53% against a Heun RK2 (1078.85 s), a 63% for a Standard RK4 (1369.17 s), a 64% for the 4p7q(6) scheme from [8] (1412.42 s), and a 77% when compared to a Heun RK3 (2167.96 s).

Note that this improvement is just considering the performance in wall clock of every simulation, and not the outcome in regards of accuracy or energy dissipation due to the integrating scheme, so that it is a comparison based on pure performance in advancing in time.

5. Conclusions

Historically bounded by the CFL condition, stability of the integration of the Navier-Stokes equations had not been revised until Trias and Lehmkuhl [6] proposed a method to determine a time-step such that it lays in the boundary of the stability region. In this paper, this idea has been extended to a set of Runge-Kutta schemes such that, based on the ratio between the biggest stable time-step and the wall clock per iteration of the scheme, as well as other additional constraints that could be defined at a pre-run stage, the most efficient scheme is used at every time, leading to a novel self-adaptive scheme.

This scheme has been tested against all the schemes that were considered when creating the pool of schemes and successful results with regards to wall clock performance have been obtained compared to the other used schemes (Tab. 1), which yields a possibility to further exploit this method so that, performancewise, better results can be obtained with a more refined programming of the method trying to reduce the computational time of the computation of this Δt. Nonetheless, it should be considered that the method would work the best when the wall clock of the computation of the time-step is negligible compared to the wall clock of the iteration computation, as the effect of having a slower Δt computation is reduced, and thus good performance should be expected for dense meshes. Nonetheless, the method has still not been tested in wall-bounded flows and thus some uncertainties with regards to the performance of the method in this kind of simulations appear.

Acknowledgements

The investigations presented in this paper are supported by the *Ministerio de Economía y Competitividad*, Spain, RETOtwin project (PDC2021-120970-I00). J.P-R. is also supported by the Catalan Agency for Management of University and Research Grants (AGAUR). The authors thankfully acknowledge these institutions.

References

[1] F. X. Trias, O. Lehmkuhl, A. Oliva, C. D. Pérez-Segarra, and R. W. Verstappen, "Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids," *Journal of Computational Physics*, vol. 258, pp. 246–267, 2014-02.

[2] A. J. Chorin, "Numerical Solution of the Navier-Stokes Equations," *Mathematics of Computation*, vol. 22, no. 104, pp. 745–762, 1968.

[3] R. W. Verstappen and A. E. Veldman, "Symmetry-preserving discretization of turbulent flow," *Journal of Computational Physics*, vol. 187, no. 1, pp. 343–368, 2003-05.

[4] B. Sanderse, "Energy-conserving Runge-Kutta methods for the incompressible Navier-Stokes equations," *Journal of Computational Physics*, vol. 233, no. 1, pp. 100–131, 2013.

[5] F. Capuano, G. Coppola, L. Rández, and L. de Luca, "Explicit Runge–Kutta schemes for incompressible flow with improved energy-conservation properties," *Journal of Computational Physics*, vol. 328, pp. 86–94, 2017-01.

[6] F. X. Trias and O. Lehmkuhl, "A self-adaptive strategy for the time integration of Navier-Stokes equations," *Numerical Heat Transfer, Part B: Fundamentals*, vol. 60, no. 2, pp. 116–134, 2011-08.

[7] B. Sanderse and B. Koren, "Accuracy analysis of explicit Runge-Kutta methods applied to the incompressible Navier-Stokes equations," *Journal of Computational Physics*, vol. 231, no. 8, pp. 3041–3063, 2012-04.

[8] F. Capuano, G. Coppola, and L. de Luca, "An efficient time advancing strategy for energy-preserving simulations," *Journal of Computational Physics*, vol. 295, pp. 209–229, 2015-08.