A SHORTCUT FOR IMEX METHODS: INTEGRATE THE RESIDUAL EXPLICITLY

SAVIO B. RODRIGUES*

Abstract. In numerical time-integration with implicit-explicit (IMEX) methods, a within-step adaptable decomposition called residual balanced decomposition is introduced. This decomposition allows any residual occurring in the implicit equation of the implicit-step to be moved into the explicit part of the decomposition. By balancing the residual, the accuracy of the local truncation error of the time-stepping method becomes independent from the accuracy by which the implicit equation is solved. In this way, the requirement of a small enough residual in an iterative solver is relieved in favor of overall computational efficiency. In order to balance the residual, the original IMEX decomposition is adjusted after the iterative solver has been stopped. For this to work, the traditional IMEX time-stepping algorithm needs to be changed. We call this new method the shortcut-IMEX (SIMEX). SIMEX can gain computational efficiency by exploring the trade-off between the computational effort placed in solving the implicit equation and the size of the numerically stable time-step. Typically, increasing the number of solver iterations increases the largest stable step-size. Both multi-step and Runge-Kutta (RK) methods are suitable for use with SIMEX. Here, we explore the efficiency of SIMEX-RK methods in overcoming parabolic stiffness. Examples of applications to linear and nonlinear reaction-advection-diffusion equations are shown. In order to define a stability region for SIMEX, a region in the complex plane is depicted by applying SIMEX to a suitable PDE model containing diffusion and dispersion. A myriad of stability regions can be reached by changing the RK tableau and the number of solver iterations.

Key words. implicit-explicit decomposition, additive Runge-Kutta, stiffness

AMS subject classifications. 65L06, 65M20, 65L04

1. Introduction.

Stiff ODEs are present in many applications, notably when the method of lines is used for the semi-discretization of PDEs. Both the parabolic stiffness due to diffusion and the hyperbolic stiffness due to the CFL condition are common sources of stiffness in PDEs. Stiffness imposes small time-step sizes in order to avoid numerical instabilities. In general, implicit time-stepping allows a larger time-step size but it requires the solution of an implicit equation at every step/stage of the method. IMEX methods decompose the right-hand-side of the ODE as the sum of two functions

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}, t) + \mathbf{g}(\mathbf{y}, t), \tag{1}$$

where $\mathbf{f}$ is the explicit part of the decomposition and $\mathbf{g}$ is the implicit part. An efficient decomposition should place non-stiff terms in $\mathbf{f}$ and stiff terms in $\mathbf{g}$ while keeping the implicit equation simple to solve. For IMEX, there is the restriction that the implicit equation needs to be solved up to a given precision in order to avoid introducing errors that could overwhelm the local truncation error; i.e., there is a small-enough-residual restriction to be fulfilled. The objective of the present article is to remove this restriction.

A new method called shortcut-IMEX (SIMEX) is introduced. It allows an iterative solver applied to the implicit equations to be interrupted at a given iteration. The reason SIMEX does not introduce errors is the use of a residual balanced decomposition: after the iterative solver is interrupted, any remaining residual of the implicit equation is accounted for in the explicit time-step by a suitable redefinition

*Universidade Federal de São Carlos, São Carlos, São Paulo 13565-905, Brazil (savio@dm.ufscar.br)

of the implicit-explicit decomposition. This within-step adjustment of the decomposition requires a modification of traditional algorithms used with IMEX. Many IMEX time-integrators can be adapted to SIMEX; namely, singly-diagonally-implicit Runge-Kutta schemes (SDIRK), explicit first stage SDIRK schemes (ESDIRK)[2, 26, 16, 22, 5, 29, 17, 27, 14, 1], and multi-step schemes [3, 23, 13, 10, 6, 28, 30, 4]. Here, we refer to IMEX-RK and IMEX-MS to distinguish Runge-Kutta (RK) methods from multi-step (MS) methods.

Recently, IMEX methods have been intensely researched in many fields of science. The choice of stiff terms being placed in $\mathbf{g}$ may vary. For example, the stiff term can be a parabolic term [20, 6, 27], or it can be a term connected to acoustic waves in the atmosphere [23, 10, 29, 14, 4, 31], or it can be a term associated to the CFL condition on a refined grid [16, 22]. In fluid mechanics, when there is parabolic stiffness, IMEX-RK allows high-order time-stepping to be used [16]. Along with discontinuous Galerkin, IMEX has been used in numerous applications [16, 27, 25, 11]. The stiffness of the CFL condition has been addressed by applying IMEX methods to the relaxation of hyperbolic equations [5]. IMEX has been compared to the deferred correction method [24] in atmospheric flows. In simulations of thermal convection in the Earth's liquid outer core, IMEX [13, 21] has been compared to exponential integrators [12] where IMEX is found to have a computational advantage.

Originally, the IMEX method was proposed in [9] as a low-order multi-step method; short after, it appeared in [8, 7] associated to *additive Runge-Kutta* (ARK) methods. An IMEX-RK method can also be called an ARK method; more precisely, it can be called an ARK_2 method, where an ARK_n method partitions the ODE in n parts. A comprehensive review of ARK methods can be found in [20, 18]. Both IMEX-RK and IMEX-MS methods can be blended into the general-linear-method (IMEX-GLM) by combining multiple stages and steps [31].

In some applications, the implicit part $\mathbf{g}$ is linearized in order to avoid solving a nonlinear system at each step [22, 14]. For example, one may rewrite the ODE as

$$\frac{d\mathbf{y}}{dt} = [\mathbf{f}(\mathbf{y}, t) + \mathbf{g}(\mathbf{y}, t) - M(\mathbf{y} - \mathbf{y}^*) - \mathbf{c}] + M(\mathbf{y} - \mathbf{y}^*) + \mathbf{c}, \tag{2}$$

and place only the linear term $M(\mathbf{y} - \mathbf{y}^*) + \mathbf{c}$ in the implicit part of the IMEX decomposition. M can be an approximation to the Jacobian derivative of $\mathbf{g}$ with respect to $\mathbf{y}$. In IMEX-RK, the choice of the decomposition can be performed at the beginning of each step [14] or remain fixed throughout time-integration [22].

In this article we explore how SIMEX overcomes parabolic stiffness using RK methods. In examples, we apply it to scalar advection-diffusion-reaction PDEs. The paper is organized as follows. A summary of IMEX-RK methods, its notation and its tableau, is found in section 2. The SIMEX-RK method is introduced in section 3 along with the algorithm; some theoretical considerations are also given in this section. Examples where SIMEX-RK is applied to ODEs are given in section 4, these ODE examples arise from a coarse spatial discretization of advection-diffusion-reaction PDEs. The stability regions of the new method is discussed in section 5. An example of application of the new method for an advection-diffusion PDE is given in section 6 where its numerical convergence with respect to space-time grid refinement is studied. Conclusions are drawn in section 7.

2. IMEX Runge-Kutta methods.

When considering IMEX, there are two large classes of time-step methods, RK methods and MS methods. In this section, the IMEX-RK methods with ESDIRK schemes are summarized.

A number of IMEX-RK schemes are available to choose from, or to be tailored to one's need. The choice of scheme may be guided, for example, by its classical order, or by the step-size-control with embedded RK-pairs, or by its economy in computational storage [15], or if it has dense-output [19]. There are schemes with all these properties within the class of ESDIRK schemes. They are commonly found in IMEX-RK methods [19, 18, 20] connected to PDE applications [2, 26, 16, 22, 5, 29, 17, 27, 14, 1]. Because the implicit equation keeps the same structure at every implicit stage, ESDIRK schemes save some computational effort by reusing Jacobians and matrix factorizations when solving implicit equations at each successive RK-stage.

An IMEX-RK method with an ESDIRK scheme consists of two specially crafted RK-methods where stages are computed in alternation, each stage "acting" exclusively either on $\mathbf{f}$ or $\mathbf{g}$. The stages are then combined at the end of each step to integrate the full ODE. Below we write the general form of a joint Butcher's tableau for an ESDIRK method with s stages:

$$\begin{array}{c|cccccc||c|cccccc}
c_1 & & & & & & & c_1 & & & & & \\
c_2 & \gamma & \gamma & & & & & c_2 & \tilde{a}_{21} & & & & \\
c_3 & a_{31} & a_{32} & \gamma & & & & c_3 & \tilde{a}_{31} & \tilde{a}_{32} & & & \\
\vdots & \vdots & \vdots & \ddots & \ddots & & & \vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \gamma & & c_s & \tilde{a}_{s1} & \tilde{a}_{s2} & \cdots & \tilde{a}_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s & & & b_1 & b_2 & \cdots & b_{s-1} & b_s
\end{array} \tag{3}$$

The empty entries are zero. The explicit method is on the right-hand side. The coefficients a_{ij} and $\tilde{a}_{ij}$ are different on each side of the tableau but the values of c_i and b_j are the same. For ESDIRK, the same element γ repeats along the diagonal, $a_{ii} = \gamma$ for $i = 2, \ldots, s$, except for a_{11} which is zero. Details about the order condition for an IMEX-RK tableau can be found in [18] and [20].

With the above tableau, the IMEX-RK algorithm is given in Algorithm 1. It describes the algorithm for a single step of size h where the input is the current approximation of $\mathbf{y}(t_n)$ denoted by $\mathbf{y}_n$. The algorithm calls a solver procedure denoted by $\mathcal{S}$. The solver $\mathcal{S}$ must return an approximate solution of the *implicit equation*,

$$\xi - h\gamma\mathbf{g}(\xi, t_n + c_i h) = \mathbf{y}_n + h\sum_{j=1}^{i-1}(a_{ij}\mathbf{k}_j + \tilde{a}_{ij}\tilde{\mathbf{k}}_j), \tag{4}$$

which must be solved for ξ.

Algorithm 1 IMEX-RK algorithm with an ESDIRK implicit scheme

```
1: $\mathbf{k}_1 = \mathbf{g}(\mathbf{y}_n, t_n)$
2: $\tilde{\mathbf{k}}_1 = \mathbf{f}(\mathbf{y}_n, t_n)$
3: for $i = 2, \ldots, s$ do
4:     $\xi = \mathcal{S}$; where $\mathcal{S}$ returns a solution of equation (4) within a tolerance $Tol$.
5:     $\mathbf{k}_i = \mathbf{g}(\xi, t_n + c_i h)$
6:     $\tilde{\mathbf{k}}_i = \mathbf{f}(\xi, t_n + c_i h)$
7: end for
8: $\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{j=1}^{s} b_j(\mathbf{k}_j + \tilde{\mathbf{k}}_j)$
9: return $\mathbf{y}_{n+1}$
```

Observe that only the right-hand side of equation (4) changes from stage to stage, this is a property of both ESDIRK and SDIRK schemes. A comment about line 5 of this algorithm: the evaluation of $\mathbf{g}$ at this line is unnecessary whenever $\mathbf{g}$ is evaluated at line 4 with the same arguments; this occurs when the residual of (4) is evaluated inside the function $\mathcal{S}$.

An example of a simple tableau that can be used with the above algorithm is the following:

$$\begin{array}{c|cc||c|cc} 0 & & & 0 & & \\ 1 & 1/2 & 1/2 & 1 & 1 & \\ \hline & 1/2 & 1/2 & & 1/2 & 1/2 \end{array} \tag{5}$$

This is a second order A-stable scheme which is a combination of Crank-Nicolson and Heun's methods henceforth called the CNH method. For CNH, Algorithm 1 is equivalent to the following formulas: given $\mathbf{y}_n$, solve for ξ the implicit equation

$$\xi - \frac{h}{2}\mathbf{g}(\xi, t_n + h) = \mathbf{y}_n + \frac{h}{2}\mathbf{g}(\mathbf{y}_n, t_n) + h\mathbf{f}(\mathbf{y}_n, t_n), \tag{6}$$

and compute

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{2}(\mathbf{g}(\mathbf{y}_n, t_n) + \mathbf{f}(\mathbf{y}_n, t_n) + \mathbf{g}(\xi, t_n + h) + \mathbf{f}(\xi, t_n + h)). \tag{7}$$

The three IMEX-RK schemes being used in this article are: (i) CNH given above, (ii) ARK548 (reference [18], page 49, labeled as ARK5(4)8L[2]SA), and (iii) ARK436 (reference [18], pages 47 and 48, labeled as ARK4(3)6L[2]SA). These two ARKpqr schemes have embedded pairs of order p and q; the number of RK-stages is r. Both ARKpqr schemes are stiffly accurate with stage order 2 where the implicit tableau is L-stable.

Motivated by the structure of Eq. (4) where both $\xi - \mathbf{y}_n$ and $\mathbf{g}(\xi, t) - \mathbf{k}_1$ are $O(h)$ for $t \in [t_n, t_n + h]$, this equation can be rewritten as

$$\eta - h\gamma(\mathbf{g}(\mathbf{y}_n + \eta, t) - \mathbf{k}_1) = \mathbf{r}, \tag{8}$$

where $\eta = \xi - \mathbf{y}_n$, $\mathbf{k}_1 = \mathbf{g}(\mathbf{y}_n, t_n)$, and the remaining terms are accounted for in the right-hand side $\mathbf{r}$. The important remark is that, in an IMEX method, the solution of Eq. (8) needs to be at least as accurate as the time-stepping method. This restriction is removed in the next section with the introduction of the SIMEX method.

3. SIMEX for Runge-Kutta methods.

In this section we define the *shortcut-IMEX* method for RK methods where ESDIRK is used in the implicit time-stepping tableau.

A SIMEX-RK method is defined with three elements: a tableau like (3), an implicit-explicit decomposition as in Eq. (1), and an *implicit-step-filter* $\mathcal{F}$ which we refer simply as a *filter*. A filter $\mathcal{F}$ can be defined, for example, from an iterative solver by stopping the iterative solver after a fixed number of iterations. Here, the distinction between a solver and a filter is necessary because neither precision nor convergence are required from a filter. Indeed, a filter $\mathcal{F}$ may yield a low precision approximation of Eq. (8). This is not a problem for SIMEX. Also, the filter can be changed from step to step but not within a step, i.e., the same filter must be used in all RK-stages belonging to a single step. We use the notation $\mathcal{F}_n$ when it is necessary to emphasize that the filter changes between steps.

The input arguments of a filter $\mathcal{F}$ are the same elements that define Eq. (8): the vectors $\mathbf{r}$ and $\mathbf{y}_n$, the real number $h\gamma$, and the function $\mathbf{g}$. The output of $\mathcal{F}$ is a vector with the same dimensions as η. As a mathematical function, we use the notation $\mathcal{F}(\mathbf{r}, \mathbf{y}_n, h\gamma; \mathbf{g}(\cdot, t))$ and we simplify the notation to $\mathcal{F}(\mathbf{r}, t)$ when some arguments are constant. Notice that $\mathcal{F}(\mathbf{r}, t)$ depends on time according to the dependence on $\mathbf{g}(\cdot, t)$ only.

In SIMEX, time-step precision is obtained by the residual balanced decomposition which we explain next. Henceforth we call the decomposition $\mathbf{f}(\mathbf{y}, t)$, $\mathbf{g}(\mathbf{y}, t)$, of Eq. (1) as the *prototype decomposition*, or *proto-decomposition*. We introduce a new decomposition

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

which we call the *residual balanced decomposition* (RBD). RBD is formally defined by choosing

$$\mathbf{g}^{rbd}(\mathbf{y}, t) = \frac{\mathbf{y} - \mathbf{y}_n - \mathcal{F}^{-1}(\mathbf{y} - \mathbf{y}_n, t)}{h\gamma} + \mathbf{k}_1 \tag{10}$$

where $\mathcal{F}^{-1}(\cdot, t)$ is the inverse function of the filter $\mathcal{F}(\mathbf{r}, t)$ with respect to the argument $\mathbf{r}$. Also, as in Eq. (8), $\mathbf{k}_1 = \mathbf{g}(\mathbf{y}_n, t_n)$. Consequently, the definition of $\mathbf{f}^{rbd}(\mathbf{y}, t)$ must be

$$\mathbf{f}^{rbd}(\mathbf{y}, t) = \mathbf{f}(\mathbf{y}, t) + \mathbf{g}(\mathbf{y}, t) - \mathbf{g}^{rbd}(\mathbf{y}, t). \tag{11}$$

Fortunately, there is an easy way to evaluate $\mathbf{g}^{rbd}$ and $\mathbf{f}^{rbd}$ without the need of computing $\mathcal{F}^{-1}(\mathbf{y} - \mathbf{y}_n, t)$. Yet, the existence of $\mathcal{F}^{-1}$ is important for the theoretical justification of the method. A discussion about the existence of $\mathcal{F}^{-1}$ is given after the SIMEX-RK algorithm.

In order to obtain a simple way to evaluate $\mathbf{g}^{rbd}$ and $\mathbf{f}^{rbd}$, we observe that

$$\eta_{rbd} = \mathcal{F}(\mathbf{r}, t)$$

is an *exact solution* of the implicit equation for the RDB decomposition:

$$\eta_{rbd} - h\gamma(\mathbf{g}^{rbd}(\mathbf{y}_n + \eta_{rbd}, t) - \mathbf{k}_1) = \mathbf{r}. \tag{12}$$

Thus, the actual value of $\mathbf{g}^{rbd}(\mathbf{y}_n + \eta_{rbd}, t)$ can be computed from Eq. (12) by the formula

$$\mathbf{g}^{rbd}(\mathbf{y}_n + \eta_{rbd}, t) = \frac{\eta_{rbd} - \mathbf{d}}{h\gamma} \tag{13}$$

where $\mathbf{d} = \mathbf{r} - h\gamma\mathbf{k}_1$. Once $\mathbf{g}^{rbd}$ is computed using Eq. (13), $\mathbf{f}^{rbd}$ is computed using Eq. (11) with $\mathbf{y} = \mathbf{y}_n + \eta_{rbd}$. Thus, RBD provides both an exactly solvable implicit-step equation, Eq. (12), and a simple way to evaluate $\mathbf{g}^{rbd}$ by using Eq. (13).

RBD has an interpretation relating it to the original proto-decomposition. The implicit equation for the proto-decomposition, Eq. (8), can be rewritten as

$$\eta - h\gamma(\mathbf{g}^{rbd}(\mathbf{y}_n + \eta, t) - \mathbf{k}_1) = \mathbf{r} + h\gamma(\mathbf{g}(\mathbf{y}_n + \eta, t) - \mathbf{g}^{rbd}(\mathbf{y}_n + \eta, t)), \tag{14}$$

and $\eta = \eta_{rbd}$ is an approximate solution where the residual is

$$h\gamma(\mathbf{g}(\mathbf{y}_n + \eta_{rbd}, t) - \mathbf{g}^{rbd}(\mathbf{y}_n + \eta_{rbd}, t)).$$

Thus, the interpretation of the RBD is that it balances this residual of Eq. (8) by transferring it to the explicit part of the RDB decomposition, namely to Eq. (11).

Algorithm 2 shows how RBD can be implemented, this is called the SIMEX-RK algorithm; it describes a single step of size h where the current value $\mathbf{y}_n$ is an input. The decomposition $\mathbf{g}^{rbd}$ and $\mathbf{f}^{rbd}$ is computed in lines 7 and 8 of the algorithm; they follow from Eqs. (13) and (11) respectively.

Algorithm 2 SIMEX-RK algorithm with an ESDIRK implicit scheme

```
1: $\mathbf{k}_1 = \mathbf{g}(\mathbf{y}_n, t_n)$
2: $\tilde{\mathbf{k}}_1 = \mathbf{f}(\mathbf{y}_n, t_n)$
3: for $i = 2, \dots, s$ do
4:     $\mathbf{d} = h \sum_{j=1}^{i-1} (a_{ij}\mathbf{k}_j + \tilde{a}_{ij}\tilde{\mathbf{k}}_j)$
5:     $\mathbf{r} = \mathbf{d} + h\gamma\mathbf{k}_1$
6:     $\eta_{rbd} = \mathcal{F}(\mathbf{r}, \mathbf{y}_n, h\gamma; \mathbf{g}(\cdot, t_n + c_i h))$
7:     $\mathbf{k}_i = (\eta_{rbd} - \mathbf{d})/(h\gamma)$
8:     $\tilde{\mathbf{k}}_i = \mathbf{f}(\mathbf{y}_n + \eta_{rbd}) + \mathbf{g}(\mathbf{y}_n + \eta_{rbd}) - \mathbf{k}_i$
9: end for
10: $\mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{j=1}^{s} b_j(\mathbf{k}_j + \tilde{\mathbf{k}}_j)$
11: return $\mathbf{y}_{n+1}$
```

A few remarks about the filter $\mathcal{F}$: Observe that $\mathcal{F}(0, t) = 0$ should be true for all filters, it is unreasonable to assume otherwise. Also, observe that if no information about Eq. (8) is used, then the choice of filter should be the identity $\mathcal{F}(\mathbf{r}, t) = \mathbf{r}$; henceforth this is called the *default-filter*. Of course, the default-filter does not help to improve numerical stability, it guarantees the "best/easiest" inversion $\mathcal{F}^{-1}$. In contrast, a filter that exactly solves Eq. (8) is called an *exact-filter*. When there is a square matrix $F(t)$ such that $\mathcal{F}(\mathbf{r}, t) = F(t)\mathbf{r}$, the filter is called linear; otherwise it is called non-linear. A linear filter can be used when Eq. (8) is nonlinear also. For example, the filter can be defined as a single Newton-like iteration applied to Eq. (8). When $\mathcal{F}$ is linear, RBD recasts the proto-decomposition of Eq. (1) into the decomposition of Eq. (2) where M and $\mathbf{c}$ are given by Eq. (10).

Even though $\mathcal{F}^{-1}$ is not directly evaluated in Algorithm 2, a proof of its existence implies that RBD is a valid IMEX decomposition for Eq. (1). Such proof places Algorithm 2 in the same theoretical grounds as Algorithm 1. Some heuristic aspects of the existence and smoothness of $\mathcal{F}^{-1}$ are discussed in the remainder of this section. We argue that these properties are easily obtainable when $\mathcal{F}$ is derived from the usual range of iterative solvers using a small number of iterations. In this discussion, assume $\mathbf{g}$ does not depend on t; thus t can be removed from the argument of all functions. Denote the function on the left-hand side of Eq. (8) by $\mathcal{H}$, $\mathcal{H}(\eta) = \eta - h\gamma\mathbf{g}(\mathbf{y}_n + \eta)$, and denote its inverse by $\mathcal{H}^{-1}$. Also, denote the Jacobian $\partial_{\mathbf{y}}\mathbf{g}(\mathbf{y}_n)$ by A, this leads to $\partial_\eta\mathcal{H}(0) = I - h\gamma A$ and $\partial_{\mathbf{r}}\mathcal{H}^{-1}(0) = (I - h\gamma A)^{-1}$.

First, consider the case where $\mathcal{F}(\mathbf{r}) = F\mathbf{r}$ is a linear filter. If $\eta_{rbd} = F\mathbf{r}$ is an approximate solution of Eq. (8) for any $\mathbf{r}$ with small norm, then one should expect the matrix norm $||F - (I - h\gamma A)^{-1}||$ to be small. This expresses that F is close to the non-singular matrix $(I - h\gamma A)^{-1}$, thus F should be non-singular too. Even if $\eta_{rbd} = F\mathbf{r}$ is merely a rough approximation to the solution of Eq. (8), it is still reasonable to expect that F should have an inverse. This is specially clear when h is small because then $(I - h\gamma A)^{-1} \approx I$. This heuristic argument can be quantified in terms of matrix norms; for example, assume that F is close enough to $(I - h\gamma A)^{-1}$

so that

$$||F - (I - h\gamma A)^{-1}|| < ||I - h\gamma A||^{-1}. \tag{15}$$

Then F is nonsingular because, given an arbitrary unitary vector $\mathbf{z}$, it follows that

$$\begin{aligned} ||F\mathbf{z}|| \geq & ||(I - h\gamma A)^{-1}\mathbf{z}|| - ||F\mathbf{z} - (I - h\gamma A)^{-1}\mathbf{z}|| \\ & \geq ||I - h\gamma A||^{-1} - ||F - (I - h\gamma A)^{-1}|| > 0. \end{aligned}$$

In the limit as h approaches zero, inequality (15) reduces to $||F - I|| < 1$ which is particularly easy to satisfy.

Second, when $\mathcal{F}$ is a *nonlinear filter*, the discussion about the existence and smoothness of $\mathcal{F}^{-1}$ becomes more complex. For instance, observe that even if $\mathcal{F}(\mathbf{r})$ is very close to $\mathcal{H}^{-1}(\mathbf{r})$ this does not guarantee that $\partial_r\mathcal{F}(\mathbf{r})$ is close to $\partial_r\mathcal{H}^{-1}(\mathbf{r})$; i.e., one cannot directly conclude that $\partial_r\mathcal{F}(0)$ is close to the non-singular matrix $(I - h\gamma A)^{-1}$. Thus, unlike linear filters, the existence of $\mathcal{F}^{-1}$ may depend on bounds of the second derivative $\partial_r^2\mathcal{F}$. Nevertheless, if one has a reason to believe that $\mathcal{F}$ is sufficiently close to a linear filter, then the existence of $\mathcal{F}^{-1}$ should be expected without apprehension. Another aspect of nonlinear filters is the smoothness of $\mathcal{F}^{-1}$. It needs to have at least as many continuous derivatives as the derivatives necessary to bound the local truncation error of the RK-method being used. From the inverse function theorem, it is known that $\mathcal{F}^{-1}$ has as many continuous derivatives as $\mathcal{F}$. When $\mathcal{F}$ is computed by subsequent iterations of Newton's method, the Jacobian of one iteration uses the answer of a previous iteration as argument; thus, the order of differentiation with respect to $\mathbf{r}$ increases at every iteration. Consequently, the regularity of $\mathcal{F}$ and $\mathcal{F}^{-1}$ could have less continuous derivatives than $\mathbf{g}$. Here, we explore in section 4 an example of nonlinear filter where things work nicely for SIMEX-RK even though we do not verify if $\mathcal{F}$ is either invertible or smooth.

The last remark about Algorithm 2 is that the filter can be chosen from a sequence of filters, $\mathcal{F}^{(1)}$, $\mathcal{F}^{(2)}$, $\mathcal{F}^{(3)}$, $\cdots$, according to a *stabilization criterion*. This choice is made at the first execution of line number 6 in Algorithm 2, i.e., when $i = 2$. Once chosen, the same filter must be used for $i = 3, \dots, s$. This is summarized in Algorithm 3 which can replace line 6 in Algorithm 2:

Algorithm 3 Filter selection according to a stabilization criterion in SIMEX-RK

```
1: if $i = 2$ then
2:     m=0
3:     repeat
4:         m=m+1
5:         $\eta_{rbd} = \mathcal{F}^{(m)}(\mathbf{r}, \mathbf{y}_n, h\gamma; \mathbf{g}(\cdot, t_n + c_i h))$
6:     until the stabilization criterion is satisfied
7: else
8:     $\eta_{rbd} = \mathcal{F}^{(m)}(\mathbf{r}, \mathbf{y}_n, h\gamma; \mathbf{g}(\cdot, t_n + c_i h))$
9: end if
```

The sequence of filters can be defined by the successive iterations of an iterative solver. Here, we do not explore Algorithm 3 in numerical experiments; instead, we experiment with each filter individually in order to understand its properties. Nevertheless, it is clear that Algorithm 2 produces almost the same calculations as Algorithm 1 whenever the stabilization criterion is chosen equal to the stopping criterion used within the iterative solver $\mathcal{S}$.

4. SIMEX for ODEs. Time-step convergence is studied in this section using examples of ODEs that originate from the semi-discretization of PDEs. The purpose of these examples is to show that SIMEX-RK can maintain accuracy even if the computational effort placed in the filter $\mathcal{F}$ is small compared to the effort placed in the solver $\mathcal{S}$ of IMEX-RK. In this section, the numerical convergence rate is studied as h is decreased while keeping the spatial discretization parameter δx fixed.

For the *first example*, we consider a 1D forced heat-equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \phi(x,t)$$

where u has zero Dirichlet boundary conditions in $x \in [0,\pi]$ and ϕ is the forcing term such that the exact solution is

$$u(x,t) = \sin(x)\sin(3x - 6\pi t). \tag{16}$$

The equation is solved by the method of lines where the spatial derivative is discretized by second-order finite differences at the points x_j which are δx apart. The resulting ODE is

$$\frac{d\mathbf{y}}{dt} = L_{\delta x}\mathbf{y} + \phi_{\delta x}(t) \tag{17}$$

where $L_{\delta x}$ is the differentiation matrix for the second derivative and the forcing term is a time-dependent vector $\phi_{\delta x}(t) = [\phi(x_1,t)\cdots\phi(x_n,t)]^t$. The proto-decomposition is chosen with $\mathbf{g}(\mathbf{y}) = L_{\delta x}\mathbf{y}$ and $\mathbf{f}(t) = \phi_{\delta x}(t)$.

Let $H_{\delta x} = I - \gamma h L_{\delta x}$ denote the matrix of the linear system found in Eq. (8), this matrix is tridiagonal. Denote the diagonal of $H_{\delta x}$ by $D_{\delta x}$ and denote the remaining part by $R_{\delta x} = H_{\delta x} - D_{\delta x}$. The filters used in SIMEX-RK are computed from Jacobi iterations. The filter $\mathcal{F}^{J(N)}(\mathbf{r})$ is defined as the N-th element $\eta^{(N)}$ of the sequence $\eta^{(i+1)} = D_{\delta x}^{-1}(-R_{\delta x}\eta^{(i)} + \mathbf{r})$ where $\eta^{(0)} = \mathbf{r}$.

In the numerical experiment shown in Figure 1, the solutions of Eq. (17) are computed using Algorithm 2, each solution with a different filter $\mathcal{F}^{J(N)}(\mathbf{r})$, $N = 0, 1, 2, 3$. In this example, Eq. (17) has 9 degrees of freedom and $\delta x = \pi/10$; this is a low-dimensional ODE which is not particularly stiff. The tableau used is the fifth order ARK548 scheme. The reference exact solution for Eq. (17) is obtained from the "ode45" routine in Matlab with absolute and relative tolerances set to 3×10^{-14}.

The numerical convergence rate for the SIMEX-RK method is *fifth order for all filters*; this is shown in Figure 1(a). Regardless the number of iterations, the SIMEX-RK method maintains the correct convergence rate. The numerical convergence of IMEX-RK in Algorithm 1 was carried out by replacing the solver $\mathcal{S}$ (line 4 of the algorithm) with each one of the filters $\mathcal{F}^{J(N)}(\mathbf{r})$; the result is shown in Figure 1(b). IMEX-RK needs at least three Jacobi iterations in order to maintain fifth order convergence.

The *second example* uses nonlinear filters. The advection-reaction-diffusion equation,

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \frac{\partial^2 u}{\partial x^2} + (1.1 - u^2)u + \psi(x,t), \tag{18}$$

is defined with ψ compatible with the exact solution (16). As in the first example, this equation also has zero Dirichlet boundary conditions in the same domain and it

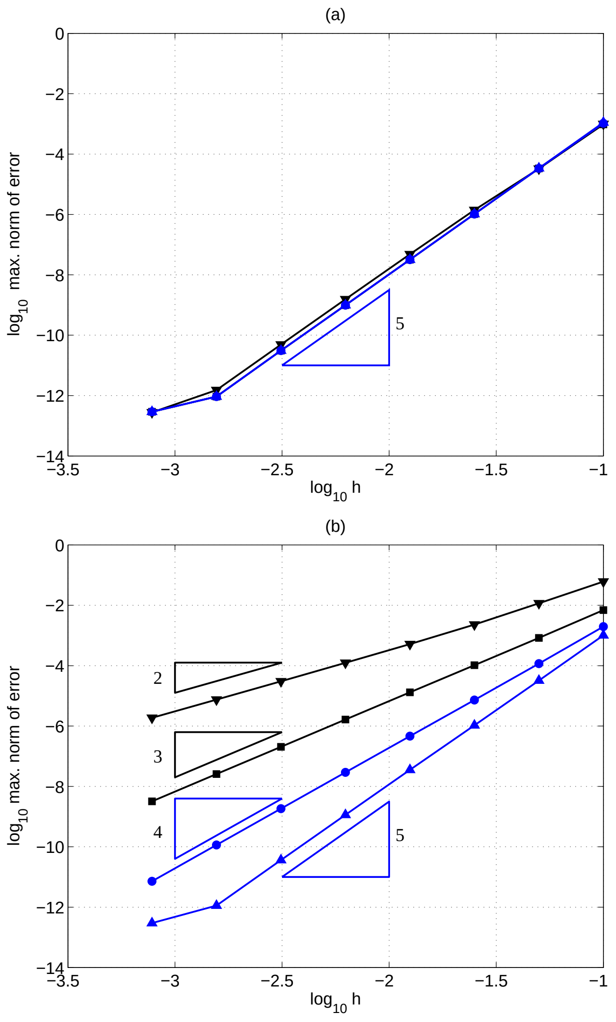

FIG. 1. *These plots show the numerical convergence rate for SIMEX-RK, part (a), and IMEX-RK, part (b). Each marker is a data point. The ordinate marks the* $\log_{10} ||\cdot||_\infty$ *of the error between the reference exact solution of Eq. (17) at* $t = 1$ *and the approximate solution with step-size* h*. The type of marker indicates the number of Jacobi iterations of the filter* $\mathcal{F}^{J(N)}$*: Markers with triangle pointing down, square, circle, and triangle pointing up, are respectively matched to 0, 1, 2, and 3, iterations. Reference slopes with value 2, 3, 4, and 5 are shown beside each line. In part (a), markers either overlap or almost overlap because SIMEX-RK is fifth order for every filter.*

is discretized using second order finite difference with $\delta x = \pi/10$ (centered difference is used for the first derivative). The discretization leads to the ODE system

$$\frac{d\mathbf{y}}{dt} = L_{\delta x}\mathbf{y} + \mathcal{N}_{\delta x}(\mathbf{y}) + \psi_{\delta x}(t) \tag{19}$$

where $\mathcal{N}_{\delta x}(\mathbf{y})$ represents the non-linear terms and $\psi_{\delta x}(t) = [\psi(x_1, t) \cdots \psi(x_n, t)]^t$ is the time-dependent forcing term. The reference exact solution of this ODE system is obtained with high precision using the "ode45" routine in Matlab.

In this example, the proto-decomposition is chosen with

$$\mathbf{g}(\mathbf{y}) = L_{\delta x}\mathbf{y} + \mathcal{N}_{\delta x}(\mathbf{y})$$

and with $\mathbf{f}(\mathbf{y}, t) = \psi_{\delta x}(t)$. We remark that this choice of $\mathbf{g}$ may not be the most efficient; nevertheless, this numerical example addresses the possible scenario where $\mathbf{g}$ is nonlinear because the stiff term is embedded within a non-linear term and, therefore, not easily identifiable. The tableau used in this example is the ARK548 also. The implicit equation, Eq. (8), is nonlinear and Newton's method is used. The filter $\mathcal{F}^{Newt(N)}(\mathbf{r})$ is defined as the element $\eta^{(N)}$ of the sequence generated by Newton's method where $\eta^{(0)} = \mathbf{r}$ followed by

$$\eta^{(i+1)} = \eta^{(i)} - [\partial_\eta \mathcal{H}(\eta^{(i)})]^{-1}(\mathcal{H}(\eta^{(i)}) + h\gamma\mathbf{k}_1 - \mathbf{r}),$$

and where, as before, $\mathcal{H}(\eta) = \eta - h\gamma\mathbf{g}(\mathbf{y}_n + \eta)$. This is a standard Newton iteration where Jacobians are computed exactly and the linear system is solved with LU decomposition.

The numerical convergence rate is shown in Figure 2. The convergence of SIMEX-RK is unaffected by the choice of filter $\mathcal{F}^{Newt(N)}$; in contrast, IMEX-RK needs either $\mathcal{F}^{Newt(2)}$ or $\mathcal{F}^{Newt(3)}$ to maintain time-step precision.

For the *third example*, we perform a numerical experiment to show that it is necessary to have a well defined smooth filter in order for SIMEX-RK to work properly (this third example could be called a "counter example" instead). In this experiment, the setting is the same used for Eq. (17) (shown in Figure 1) the only exception being the definition of the filter. A "purposely bad filter" $\mathcal{F}^{bad}(\mathbf{r})$ is defined as being either equal to $\mathcal{F}^{J(2)}(\mathbf{r})$ when i is even in Algorithm 2 or equal to $\mathcal{F}^{J(3)}(\mathbf{r})$ when i is odd. This means that either 2 or 3 Jacobi iterations are being alternately used to approximate the solution of the linear system at each RK-stage. The result of this numerical experiment (not shown graphically) is the following: the convergence rate of SIMEX-RK drops to fourth order instead of maintaining the fifth order shown in Figure 1(a). It is the RK-stage alternation that causes the loss of precision. For completion, consider the following experiment: in Algorithm 2, use an "ok" filter $\mathcal{F}^{ok}_n(\mathbf{r})$ where $\mathcal{F}^{ok}_n$ is either equal to $\mathcal{F}^{J(2)}$ when n is even or $\mathcal{F}^{J(3)}$ when n is odd; i.e., the number of Jacobi iterations alternates *between steps* rather than between RK-stages. The result of this experiment (not shown graphically) is that SIMEX-RK converges with fifth order just as shown in Figure 1(a). In summary, if a filter is comprised of an iterative solver then the number of iterations of the iterative solver must remain fixed at every RK-stage of each time-step.

5. Time-step stability of SIMEX for PDEs.

In this section we explore time-step stability of the SIMEX-RK algorithm when applied to a stiff model PDE. The numerical stability is modified by the choice of tableau and filter.

For methods that do not decompose the ODE, i.e. monolithic methods, the method's stability can be described by a region in the complex plane. The *scalar model equation*, $y' = zy$ with $y(0) = 1$ and $z \in \mathbb{C}$, is used in order to depict the region of $\mathbb{C}$ where the method yields an asymptotically decreasing sequence of values as time-steps progress. The analysis of this scalar model equation can be justified using the canonical form of a general linear ODE system with constant coefficients.

For SIMEX methods, the stability depends on many things: beside the choice of the IMEX tableau and the choice of the proto-decomposition, the stability depends on the filter also. The scalar model equation is not useful for studying SIMEX stability

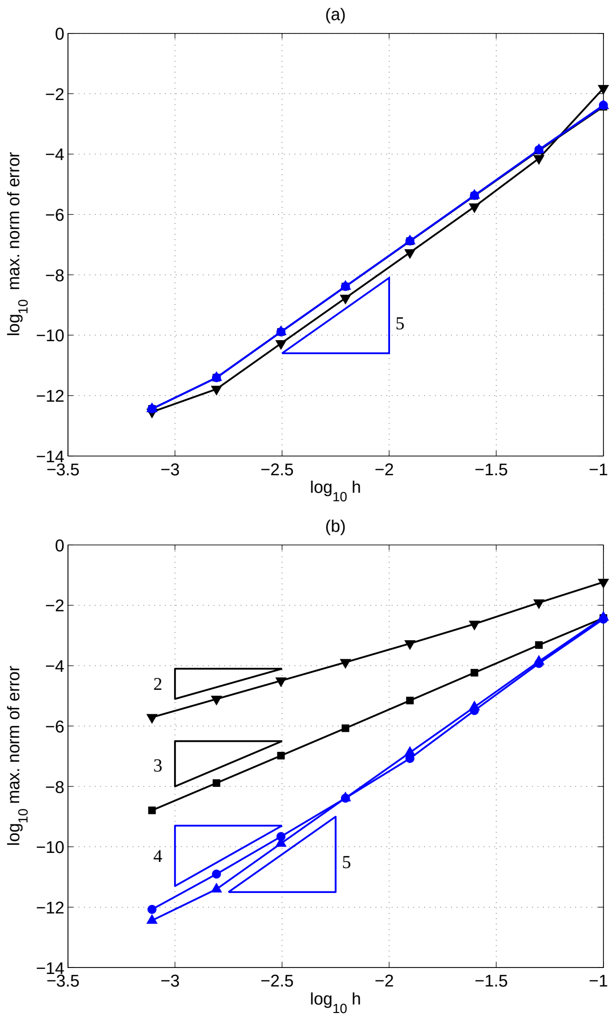

FIG. 2. *These plots show the numerical convergence of SIMEX-RK, part (a), and IMEX-RK, part (b) when applied to Eq. (19). The axis are the same as in Figure 1. The type of marker indicates the number of iterations of Newton's method: the markers with triangle pointing down, square, circle, and triangle pointing up, are respectively matched to the filter $\mathcal{F}^{Newt(N)}$ with N = 0, 1, 2, and 3. In part (a), the markers almost overlap: SIMEX-RK is fifth order for every N. In part (b), the convergence rate depends on the filter. When $N = 3$, IMEX-RK is certainly fifth order but when $N = 2$ the numerical convergence rate is closer to fourth order.*

because the influence of the filter is lost. Here we propose a generalization of the scalar model equation by considering the stability of the SIMEX method for an ODE system with the form

$$\frac{d\mathbf{y}}{dt} = zA\mathbf{y}, \tag{20}$$

where $z \in \mathbb{C}$ and A is a matrix with one eigenvalue is equal to 1 and $\rho(A) = 1$ (spectral radius). The *stability region* $\Omega(A)$ is defined as the subset of $\mathbb{C}$ such that $z \in \Omega(A)$ if, and only if, the numerical sequence $\mathbf{y}_n$ generated by the time-stepping method applied to Eq. (20) whith $h = 1$ converges asymptotically to zero as $n \to +\infty$ for every unit-size initial condition $\|\mathbf{y}_0\|_\infty = 1$. With this definition, the region defined by the scalar model equation is equal to $\Omega(1)$.

The matrix A should represent a typical application of interest. Here we consider the equation

$$\frac{\partial u}{\partial t} = -\lambda \Delta u \tag{21}$$

where $\lambda \in \mathbb{C}$, and where $u(\mathbf{x}, t)$ is defined in the domain $\mathbf{x} \in [0, \pi]^2$ with periodic boundaries. This is a suitable model for addressing both diffusive and dispersive second order PDEs. The solutions are asymptoticaly stable when $\Re(\lambda) < 0$ and oscillatory when λ is imaginary. The discretized Laplacian operator $\Delta_{\delta x}$ is obtained with the standard second-order, five points, finite-difference stencil on a uniform grid. Let N denote the number of discretization points along each edge of the domain and let $\delta x = \pi/N$ be the spacing between grid points. We define the matrix A_N from the discretized Laplacian according to $A_N = -8^{-1}(\delta x)^2 \Delta_{\delta x}$. This multiplicative constant ensures the eigenvalues of A_N are within the interval $(0, 1)$; moreover, the largest eigenvalue of A_N approaches the value 1 with the rate $O(N^{-2})$. For example, the largest eigenvalue of A_{50} is 0.999. For the purpose of depicting the stability region, we can simply set $A = A_N$ in Eq. (20).

Henceforth, we study the stability of SIMEX-RK methods by applying it to Eq. (20) with the matrix $A = A_N$. We choose the proto-decomposition of Eq. (20) with everything implicit: $\mathbf{g} = zA_N\mathbf{y}$, and $\mathbf{f} = 0$. When comparing Eq. (20) with the discretized version of Eq. (21), z is related to λ according to the formula

$$z = \frac{8\lambda}{\delta_x^2}. \tag{22}$$

By changing the filter and the tableau, the stability region $\Omega(A_N)$ attains different sizes and shapes; this is shown in the figures that follow. These regions are computed by placing a grid on the complex-plane and then applying the algorithm with $h = 1$ for up to 30 steps for each value of z on the grid. A random initial condition is chosen and, after 30 steps, the amplification factor of the last step is stored. The resulting data is then smoothed with a low-pass filter and the level curve of value 1 is plotted as the boundary of the region. For values of z outside each depicted region, a random initial condition becomes larger as the steps increase. The regions shown here are obtained with $N = 50$. When using $N = 100$ or larger, changes in these regions are very small.

Figure 3 shows stability regions obtained using SIMEX-RK with the CNH tableau and Figure 4 using the ARK436 tableau. The filters used in these figures are the following: (i) $\mathcal{F}^I$ is the default-filter. (ii) $\mathcal{F}^{ATS(1)}$ computes one iteration of an alternate-direction tridiagonal solver (ATS). (iii) $\mathcal{F}^{GS(5)}$ applies five Gauss-Seidel iterations. (iv) $\mathcal{F}^{ATS(3)}$ computes 3 iterations of ATS. (v) $\mathcal{F}^{ILU}$ approximately solves the linear system with an incomplete-LU factorization (ILU). (vi) $\mathcal{F}^{SR(5)}$ computes 5 successively-relaxed GS iterations. The parameters defining these filters are detailed next.

Each ATS iteration used in $\mathcal{F}^{ATS(\cdot)}$ solves two tridiagonal systems. The matrix H of the linear system is decomposed as $H = T - E$ where T is tridiagonal. Then, an ATS iteration is

$$T\mathbf{x}^{(k+1/2)} = E\mathbf{x}^{(k)} + \mathbf{r}, \quad T\tilde{\mathbf{x}}^{(k+1)} = E\tilde{\mathbf{x}}^{(k+1/2)} + \mathbf{r},$$

where $\tilde{\mathbf{x}}$ is the permutation of $\mathbf{x}$ produced by interchanging the roles of the horizontal and the vertical directions in the natural ordering of nodes.

The ILU filter, $\mathcal{F}^{ILU}$, approximately solves the linear system $H\mathbf{x} = \mathbf{r}$ with an incomplete-LU factorization using the "ilu" routine in Matlab; the drop-tolerance is set to 0.02. This means any element less than 2% the size of the pivot is dropped.

Each iteration in $\mathcal{F}^{SR(\cdot)}$ "relaxes" one GS iteration where the relaxation parameter is set to 0.9; this is actually an "under-relaxation". This value of the relaxation parameter was tested against other values and it produced the largest stable interval along the real axis in the case of the CNH tableau. Different from other filters, $\mathcal{F}^{SR(5)}$ is not based on a convergent linear solver. This filter has a bigger stability region than $\mathcal{F}^{GS(5)}$ in the case of the CNH tableau but both filters have similar stability regions in the case of ARK436.

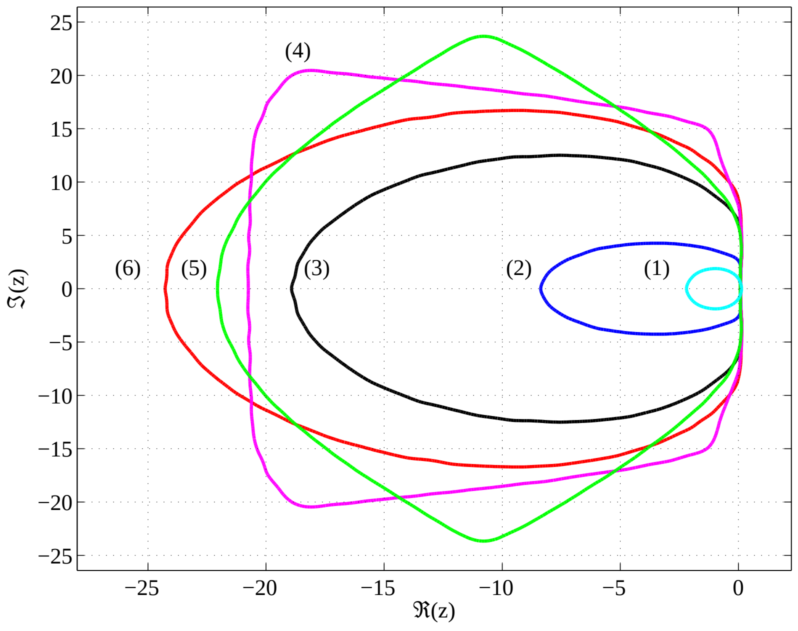

FIG. 3. *Complex-plane showing stability regions of SIMEX-RK generated with the CNH tableau, cf. (5), for six different filters which result in the regions numbered (1) through (6). The corresponding filters are: (1) $\mathcal{F}^{I}$, (2) $\mathcal{F}^{ATS(1)}$, (3) $\mathcal{F}^{GS(5)}$, (4) $\mathcal{F}^{ATS(3)}$, (5) $\mathcal{F}^{ILU}$, (6) $\mathcal{F}^{SR(5)}$; the definition of these filters is in the text. For the exact-filter, the stability region is the same as the Crank-Nicolson scheme: the left-hand side of the complex-plane.*

Several interesting features can be observed in Figs. 3 and 4. Primarily, it is visible that ARK436 obtains a larger stability region than CNH for the same filter. Of course, the computational effort placed in each step is different; ARK436 has five implicit RK-stages while CNH has only one. The $\mathcal{F}^{ILU}$ filter, region (5), produces a remarkably large stability region in ARK436; it reaches up to $z = -644$ in the real axis. We should remark that $\mathcal{F}^{ILU}$ is not a constant-effort filter; the fill-in rate changes as z changes because the drop-tolerance is being kept fixed in the ILU decomposition. Nevertheless, the fill-in at $z = -644$ is modest when compared with the matrix of the linear system H: the number of non-zero entries in the lower-triangular matrix L is 0.98 times the number of non-zero entries in H.

Another interesting feature is that a segment of the imaginary axis is inside the

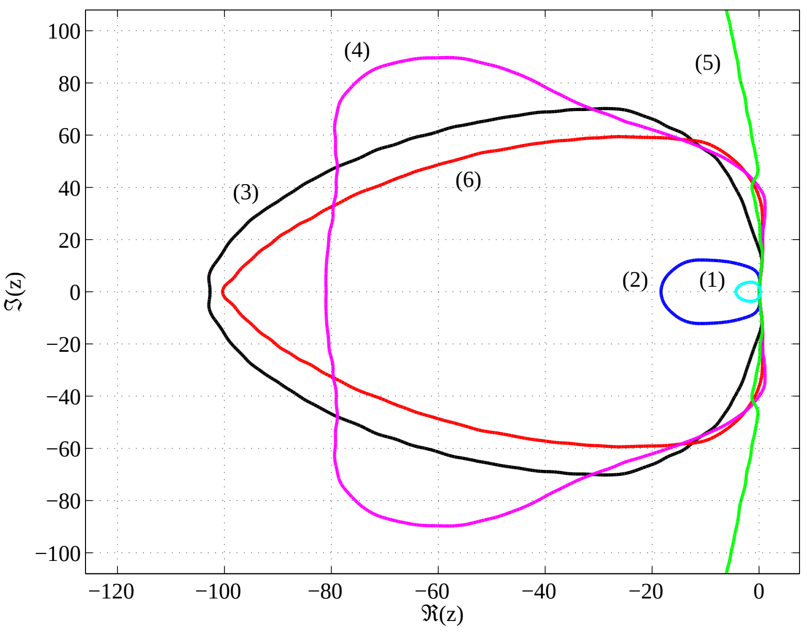

FIG. 4. *Complex-plane showing stability regions of SIMEX-RK generated with the ARK436 tableau. The filters used are the same as in Figure 3. The stability region computed with $\mathcal{F}^{ILU}$ is a closed region, region (5), but it extends beyond this frame; its boundary crosses the real axis at $\Re(z) = -644$.*

stability region for some filters in Figure 4. For instance, the segment $[-16i, 16i]$ is contained in the stability region of the filter $\mathcal{F}^{ILU}$. Moreover, the regions associated with filters $\mathcal{F}^{GS(5)}$ and $\mathcal{F}^{SR(5)}$ contain the segment $[-35i, 35i]$. This may be a consequence of the L-stable implicit scheme used in ARK436.

In SIMEX-RK the size of the stability region is largely controlled by the choice of the filter. Combinations of tableau and filter pairs provide an immense range of possibilities to explore. The most appropriate combination depends on the particular PDE and on the precision to be attained; Eq. (21) should be viewed as a standard example where filters can be compared.

6. Space-time grid convergence for a linear advection-diffusion PDE. This section shows how SIMEX-RK can be used for the solution of an advection-diffusion equation in 2D where parabolic stiffness is predominant as the grid gets refined. Parabolic PDEs require ever larger stability regions when δ_x is being reduced while keeping the ratio h/δ_x fixed.

Here, we obtain numerical solutions of

$$\frac{\partial u}{\partial t} + \mathbf{v} \cdot \nabla u = 0.3\Delta u + \psi \tag{23}$$

where $u(x_1, x_2, t)$ is defined in the spatial domain $[0, \pi]^2$ with periodic boundaries and where $\mathbf{v} = (1/2, \sqrt{3}/2)$ is a constant unitary vector. The r.h.s. $\psi(t, x_1, x_2)$ is consistent with the exact solution $u_{exact}(t, x_1, x_2) = \exp(-\sin(t - 4x_1 - 2x_2))$. All partial derivatives are discretized with fourth-order finite-difference formulas which use stencils with five points in each Cartesian direction. In this way, the discretized Laplacian has a 9-point stencil with a cross shape. As before, N denotes the number of discretization points along each Cartesian direction; the spatial grid is uniform with δx spacing between grid points. Using the method of lines, this leads to an ODE system

$$\mathbf{y}' = \frac{1}{\delta x^2} M_{\delta x}\mathbf{y} + \frac{1}{\delta x} A_{\delta x}\mathbf{y} + \psi_{\delta x}(t) \tag{24}$$

where $\mathbf{y} \in \mathbb{R}^{N^2}$ and $M_{\delta x}$ and $A_{\delta x}$ denote the matrices resulting from the discretization of the Laplacian and of the advection term respectively; $\psi_{\delta x}(t)$ is the forcing term where ψ is evaluated at grid points. The proto-decomposition is $\mathbf{g}(\mathbf{y}) = \delta x^{-2} M_{\delta x}\mathbf{y}$ and $\mathbf{f}(\mathbf{y}, t) = \delta x^{-1} A_{\delta x} + \psi_{\delta x}(t)$. At every implicit RK-stage, the matrix of the linear system $I - (h/\delta x^2)M_{\delta x}$ has $9N^2$ nonzero elements. For time-integration, we use the fourth order ARK436 tableau so that the resulting scheme is both fourth order in time and space.

Numerical results are shown in Figure 5 for seven different space-time grids: $h_j = 2^{-j}/5$; for $j = 1, \ldots, 7$. The ratio of time-step size to grid-space is kept fixed, $h_j/\delta x_j = \pi^{-1}$. Ten different filters were tested with SIMEX-RK, these filters can be grouped in two categories: (i) GS filters, denoted by $\mathcal{F}^{GS(k)}$, which perform k Gauss-Seidel iterations where $0 \leq k \leq 7$ ($k = 0$ being the default-filter); and (ii) preconditioned Conjugate Gradient Squared (CGS) filters, denoted by $\mathcal{F}^{CGS(p)}$, which apply p iterations of the CGS method using an ILU decomposition as preconditioner. The drop-tolerance for the ILU decomposition is 0.02. The number of CGS iterations is either $p = 1$ or $p = 2$. The fill-in is small when using this drop-tolerance: at the finest grid the number of nonzero elements in L and in U divided by $9N^2$ is equal to 1.11; this means an 11% increase in memory use.

Figure 5 shows the numerical error of the best filter for SIMEX-RK at each grid. The best filter is the one with the smallest computational time among all filters with stable time-integration up to $t = 1$. In this experiment, numerical instability is detected when any component of the numerical solution exceeds 10^3 at any time-step along the algorithm. SIMEX-RK data points are marked with a plus sign (+) and joined by a dashed line. A line with slope 4 is placed in the diagonal of this figure. SIMEX-RK's errors follow closely the fourth order convergence rate.

For comparison, Figure 5 brings the error of the "best filter" for IMEX-RK also. The data points are marked with circles ($\circ$). This data is obtained replacing the solver in IMEX-RK by each one of the filters. As before, the best filter is the one with the least computational time among the filters with stable time-integration. Of course, IMEX-RK may become imprecise because each filter has a fixed number of iterations. Nevertheless, using this criterion we can distinguish whether a filter used in SIMEX-RK is solving Eq. (8) accurately or not.

SIMEX-RK reaches the correct numerical convergence rate even though Eq. (8) is not being accurately solved at some of these grids. At grids h_4 and h_7 of Figure 5, SIMEX-RK is more precise than IMEX-RK while using the same number of iterations. This advantage can be attributed to the extra precision provided by RBD. Furthermore, at grids h_5 and h_6, IMEX-RK needs more iterations than SIMEX-RK in order to reach a stable, albeit imprecise, solution. It may be a fortunate coincidence that the improved numerical precision provided by RBD results in a stable outcome for SIMEX-RK with less iterations than for IMEX-RK.

In order to show how different filters behave, part of the data of Figure 5 is expanded in Figure 6. Each marker in Figure 6 shows the numerical error vs. CPU time for a given filter/method pair at a given grid. This figure shows the filters $\mathcal{F}^{GS(4)}$, $\mathcal{F}^{GS(6)}$, $\mathcal{F}^{CGS(1)}$, and $\mathcal{F}^{CGS(2)}$, when used with the grids h_j for $j \geq 3$. Only

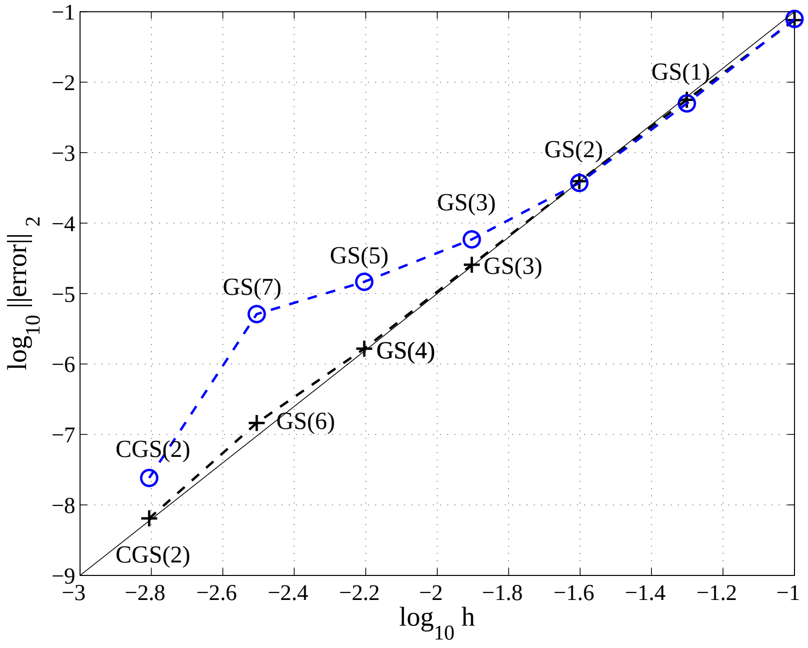

Fig. 5. *Advection-diffusion Eq.* (24) *is numerically integrated with SIMEX-RK and IMEX-RK algorithms. The horizontal axis shows the step-size h. The vertical axis shows the discrete L^2-norm of the error to the exact solution at time $t = 1$. Spatial grid points are spaced with $\delta x_j = h_j/\pi$. The markers $\circ$ and $+$ are data points for IMEX-RK and SIMEX-RK respectively. A continuous line with slope 4 is placed in the diagonal. The filter/solver used for SIMEX/IMEX is written beside each marker; for markers that overlap, the same description applies for both. At $h_1 = 10^{-1}$ the filter/solver is $GS(0)$.*

the combinations that yield stable results are shown. SIMEX-RK allows every filter to be used in more grids than IMEX-RK. In particular, at grid h_6, SIMEX-RK can reach an accurate solution using either $\mathcal{F}^{GS(6)}$ or $\mathcal{F}^{CGS(1)}$; in contrast, IMEX-RK needs to use $\mathcal{F}^{CGS(2)}$. At this grid, $\mathcal{F}^{GS(6)}$ is slightly less precise than $\mathcal{F}^{CGS(1)}$; perhaps this is a sign of borderline stability which indicates that $\mathcal{F}^{GS(6)}$ does not bear more grid refinements. At the finest grid, h_7, IMEX-RK needs to use more than two CGS iterations in order to reach an accurate solution.

We remark that, in view of Algorithm 3, any stopping criterion used in IMEX-RK can be used as a stabilization criterion in SIMEX-RK also. According to the numerical experiments shown here, the stabilization criterion used with SIMEX-RK is likely to be less strict than the stopping criterion used with IMEX-RK.

7. Conclusion. In this article the residual balanced decomposition (RBD) is introduced. Given an implicit-explicit decomposition and a filter, RBD provides a new decomposition where the implicit equation is solved exactly without additional computational work. This remarkable property allows great freedom for exploring the numerical properties of the resulting algorithm. The decomposition itself is rather abstract, Eqs. (10) and (11), but the computational steps for its evaluation are straightforward, Eq. (13). Here, RBD is used with IMEX-RK methods that have ESDIRK implicit schemes, the resulting method is the SIMEX-RK method, Algorithm 2.

One way to construct a filter is to arbitrarily fix the number of iterations of

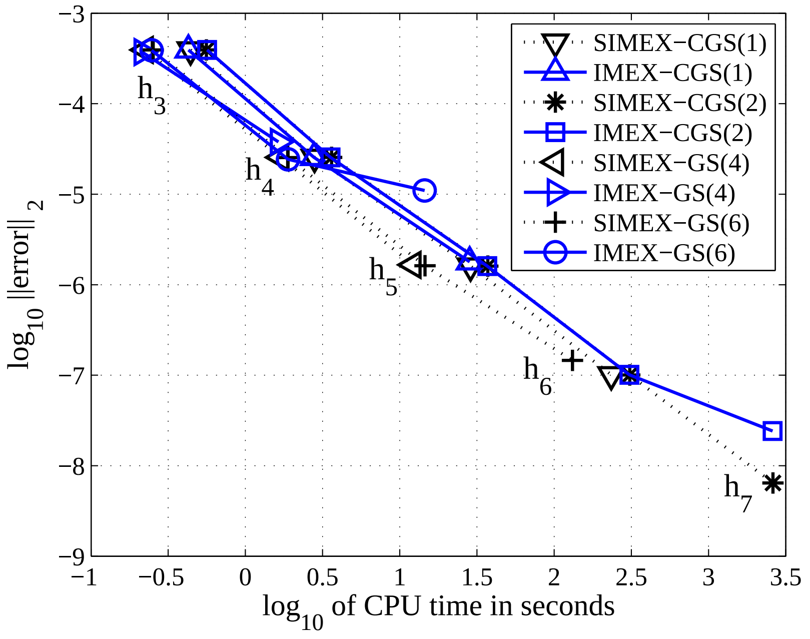

Fig. 6. *Data points show the numerical error versus the computation time (shown as* $\log_{10}$ *of the CPU time) necessary for time-integration while varying the filter, the method, and the grid-size. Dotted lines join the data computed using SIMEX-RK while solid lines join the data of IMEX-RK. Each cluster of data points corresponds to a different grid which is indicated in the vicinity of the cluster. The data points of SIMEX-RK are predominant in the lower-left boundary of the data.*

an iterative solver applied to the implicit equation of the original IMEX method, Eq. (8). Computational experiments confirm that SIMEX-RK maintains the correct numerical convergence rate even though its filter yields an inaccurate solution for Eq. (8). Thus, the goal of the filter is to ensure time-step stability and not an accurate solution of Eq. (8). Numerical experiments in section 6 show a spatial-temporal grid refinement of an advection-diffusion PDE. In these experiments SIMEX-RK can produce an accurate solution as soon as its filter is strong enough to yield a stable outcome.

In order to study the stability of the SIMEX-RK method, the stability-region in the complex-plane is generalized to ODE systems where the largest eigenvalue of the matrix A is equal to one. As shown in section 5, the stability region can vary significantly depending on the combination of RK scheme and filter being used. The size of the stability region clearly depends on the computational effort placed in the filter. This region can reach far into the negative real axis as observed with the ILU filter. Also, when the implicit RK scheme is L-stable, SIMEX-RK can produce stability regions that contain a significant interval of the imaginary axis.

SIMEX can be extended to multi-step (MS) methods. This occurs because equations (12) through (14) are valid as long as the implicit equation remains the same at each step of the MS method. A careful initialization procedure is necessary when using SIMEX-MS method (an example is available as supplementary material). RBD can be used with some types of general-linear-methods (GLM) also. One such GLM is found in a recent study [31]. The RK part of the GLM method needs to have the singly-diagonally-implicit property; such property ensures that the same implicit equation appears at every stage.

The search for performance improvements with SIMEX-RK can start with Algorithms 2 and 3 where the sequence of filters and the stabilization criterion are provided by the incumbent IMEX-RK method. At this point, it is likely that SIMEX-RK will closely replicate the incumbent IMEX-RK method. Then, SIMEX-RK may be improved by relaxing the stabilization criterion. Further improvements may result either from using less expensive preconditioners or from constructing preconditioners that target specific sub-spaces where the numerical instability is present. SIMEX also opens the possibility of using filters that are not based on iterative solvers.

REFERENCES

[1] X. Antoine, C. Besse, and V. Rispoli, *High-order imex-spectral schemes for computing the dynamics of systems of nonlinear Schrodinger/Gross-Pitaevskii equations*, J. Comp. Phys., 327 (2016), pp. 252–269.

[2] U. M. Ascher, S. J. Ruuth, and R. J. Spiteri, *Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations*, Applied Numerical Mathematics, 25 (1997), pp. 151–167.

[3] U. M. Ascher, S. J. Ruuth, and B. T. R. Wetton, *Implicit-explicit methods for time-dependent partial differential equations*, SIAM J. Numer. Anal., 32 (1995), pp. 797–823.

[4] S. Blaise, J. Lambrechts, and E. Deleersnijder, *A stabilization for three-dimensional discontinuous Galerkin discretizations applied to nonhydrostatic atmospheric simulations*, International Journal for Numerical Methods in Fluids, 81 (2016), pp. 558–585.

[5] S. Boscarino, L. Pareschi, and G. Russo, *Implicit-explicit Runge-Kutta schemes for hyperbolic systems and kinetic equations in the diffusion limit*, SIAM J. Sci. Comput., 35 (2013), pp. A22–A51.

[6] J. H. Chaudhry, D. Estep, V. Ginting, J. N. Shadid, and S. Tavener, *A posteriori error analysis of IMEX multi-step time integration methods for advection-diffusion-reaction equations*, Computer Methods in Applied Mechanics and Engineering, 285 (2015), pp. 730–751.

[7] G. J. Cooper and A. Sayfy, *Additive methods for the numerical solution of ordinary differential equations*, Math. Comp., 35 (1980), pp. 1159–1172.

[8] G. J. Cooper and A. Sayfy, *Addtive Runge-Kutta methods for stiff ordinary differential equations*, Math. Comp., 40 (1983), pp. 207–218.

[9] M. Crouzeix, *Une méthode multipas implicite-explicite pour l'approximation des équations d'evolution paraboliques*, Numer. Math., 35 (1980), pp. 257–276.

[10] D. R. Durran and P. N. Blossey, *Implicit-explicit multistep methods for fast-wave-slow-wave problems*, Monthly Weather Review, 140 (2012), pp. 1307–1325.

[11] B. Froehle and P.-O. Persson, *A high-order discontinuous Galerkin method for fluid-structure interaction with efficient implicit-explicit time stepping*, J. Comp. Phys., 272 (2014), pp. 455–470.

[12] F. Garcia, L. Bonaventura, M. Net, and J. Sánchez, *Exponential versus imex high-order time integrators for thermal convection in rotating spherical shells*, J. Comp. Phys., 264 (2014), pp. 41–54.

[13] F. Garcia, M. Net, B. García-Archilla, and J. Sánchez, *A comparison of high-order time integrators for thermal convection in rotating spherical shells*, J. Comp. Phys., 229 (2010), pp. 7997–8010.

[14] D. Ghosh and E. Constantinescu, *Semi-implicit time integration of atmospheric flows with characteristic-based flux partitioning*, SIAM Journal on Scientific Computing (SISC), 38 (2016), pp. A1848–A1875, doi:10.1137/15M1044369.

[15] I. Higueras and T. Roldán, *Construction of additive semi-implicit RungeKutta methods with low-storage requirements*, J. Sci. Comput., 67 (2016), pp. 1019–1042.

[16] A. Kanevsky, M. H. Carpenter, D. Gottlieb, and J. S. Hesthaven, *Application of implicit-explicit high order Runge-Kutta methods to discontinuous-Galekin schemes*, J. Comp. Physics, 225 (2007), pp. 1753–1781.

[17] V. Kazemi-Kamyab, A. van Zuijlen, and H. Bijl, *Analysis and application of high order implicit runge-kutta schemes for unsteady conjugate heat transfer: A strongly-coupled approach*, J. Comput. Physics, 272 (2014), pp. 471–486.

[18] C. A. Kennedy and M. H. Carpenter, *Additive Runge-Kutta schemes for convection-diffusion-reaction equations*, Technical report NASA/TM-2001-211038, (2001).

[19] C. A. Kennedy and M. H. Carpenter, *Diagonally implicit Runge-Kutta methods for ordinary differential equations. a review*, Technical report NASA/TM-2016-219173, (2016).

[20] C. A. Kennedy, M. H. Carpenter, and R. M. Lewis, *Additive Runge-Kutta schemes for convection-diffusion-reaction equations*, Appl. Numer. Math., 44 (2003), pp. 139–181.

[21] P. Mati, M. A. Calkins, and K. Julien, *A computationally efficient spectral method for modeling core dynamics*, Geochemeistry, Geophysics, Geosystems, 17 (2016), pp. 3031–3053.

[22] P.-O. Persson, *High-order les simulations using implicit-explicit runge-kutta schemes*, in 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, AIAA 2011-684, (2011), doi:10.2514/6.2011-684.

[23] M. Restelli and F. X. Giraldo, *A conservative discontinuous Galerkin semi-implicit formulation for the Navier-Stokes equations in nonhydrostatic mesoscale modeling*, SIAM J. Sci. Comput., 31 (2009), pp. 2231–2257.

[24] D. Ruprecht and R. Speck, *Spectral deferred corrections with fast-wave slow-wave splitting*, SIAM J. Sci. Comput., 38 (2016), pp. A2535–A2557.

[25] M. P. Ueckermann and P. F. J. Lermusiaux, *Hybridizable discontinuous Galerkin projection methods for Navier-Stokes and Boussinesq equations*, J. Comp. Physics, 306 (2016), pp. 390–421.

[26] A. H. van Zuijlen and H. Bijl, *Implicit and explicit higher order time integration schemes for structural dynamics and fluid-structure interaction computations*, SIAM J. Sci. Comput., 38 (2016), pp. A1430–A1453.

[27] H. Wang, C.-W. Shu, and Q. Zhang, *Stability and error estimates of local discontinuous galerkin methods with implicit-explicit time-marching for advection-diffusion problems*, SIAM J. Numer. Anal., 53 (2015), pp. 206–227.

[28] H. Wang, C.-W. Shu, and Q. Zhang, *Stability analysis and error estimates of local discontinuous galerkin methods with implicitexplicit time-marching for nonlinear convectiondiffusion problems*, Applied Mathematics and Computation, 272 (2016), pp. 237–258.

[29] H. Weller, S.-J. Lock, and N. Wood, *RungeKutta IMEX schemes for the horizontally explicit/vertically implicit (HEVI) solution of wave equations*, J. Comp. Physics, 252 (2013), pp. 365–381.

[30] M. Zayernouri and A. Matzavinos, *Fractional Adams-Bashforth/Moulton methods: An application to the fractional Keller-Segel chemotaxis system*, J. Comp. Phys., 317 (2016), pp. 1–14.

[31] H. Zhang, A. Sandu, and S. Blaise, *High order implicit-explicit general linear methods with optimized stability regions*, SIAM J. Sci. Comput., 38 (2016), pp. A1430–A1453.