Optimization of High-Order Diagonally-Implicit Runge-Kutta Methods

Pieter D. Boom[a,*], David W. Zingg[a,1]

[a]*Institute for Aerospace Studies, University of Toronto, Toronto, Ontario, M3H 5T6, Canada*

Abstract

This article presents constrained numerical optimization of high-order linearly and algebraically stable diagonally-implicit Runge-Kutta methods. After satisfying the desired order conditions, undetermined coefficients are optimized with respect to objective functions which consider accuracy, stability, and computational cost. Constraints are applied during the optimization to enforce stability properties, to ensure a well-conditioned method, and to limit the domain of the abscissa. Two promising third-order methods are derived using this approach, labelled SDIRK[3,(1,2,2)](3)L_14 and SDIRK[3,1](4)L_SA_5. Both optimized schemes have a good balance of properties. The relative error norm of the latter, the L_2-norm scaled by a function of the number of implicit stages, is a factor of two smaller than comparable methods found in the literature. Variations on these methods are discussed relative to trade-offs in their accuracy and stability properties. A novel fifth-order scheme SDIRK[5,1](5)L_02 is derived with a significantly lower relative error norm than the comparable fifth-order A-stable reference method. In addition, the optimized scheme is L-stable. The accuracy and relative efficiency of the Runge-Kutta methods are verified through numerical simulation of van der Pol's equation, as well as numerical simulation of vortex shedding in the laminar wake of a circular cylinder, and in the turbulent wake of a NACA 0012 airfoil. These results demonstrate the value of numerical optimization for selecting undetermined coefficients in the construction of high-order

*Corresponding author (+1-647-221-2587), Postdoctoral Researcher

Email address: pieter.boom@mail.utoronto.ca (Pieter D. Boom)

[1]University of Toronto Distinguished Professor of Computational Aerodynamics and Sustainable Aviation; Director, Center for Sustainable Aviation

Runge-Kutta methods with a balance between competing objectives.

Keywords: Implicit Time-Marching methods, Implicit Runge-Kutta Methods, High-Order Methods, Unconditional Stability, Optimization, Ordinary Differential Equations, Initial-Value Problems, Computational Fluid Dynamics

---

1. Introduction

Many phenomena in science and engineering are inherently time-dependent and modelled with ordinary differential equations (ODEs), either directly or through semi-discretization of partial differential equations. Consider fluid dynamics as an example, where time-dependent phenomena include: vortex shedding, transition, turbulence, and acoustics. Semi-discretization of the governing equations describing these flows, the Navier-Stokes equations, results in a large coupled system of nonlinear ODEs.

Both the physics and numerical approximation of these problems can lead to stiff ODEs [29, 57]. This is characterized by the presence of parasitic modes in addition to the relevant driving modes. These parasitic modes do not influence the accuracy of numerical solution but must remain stable [57]. Therefore, efficient numerical simulation of stiff ODEs requires a solution process that balances the accuracy of the driving modes, the stability of the parasitic modes, and computational cost.

Unconditionally stable implicit time-marching methods enable the selection of the time step size based on the desired accuracy of the driving modes [30]. In contrast, conditionally stable methods require that the time step size be chosen to ensure the stability of the parasitic modes. For the simulation of stiff ODEs, the maximum stable time step size is significantly smaller than required to attain the desired accuracy. This translates into an increased number of time steps, which correlates with the stiffness of the ODE being solved. The reduction in number of time steps enabled by unconditionally stable implicit methods can outweigh the increased computational cost associated with an implicit process [29]. As a result, unconditionally stable implicit time-marching methods are a common choice for solving stiff ODEs.

The focus of this article is on the class of high-order unconditionally stable implicit Runge-Kutta methods. While low-order methods can obtain an arbitrary level of accuracy, the number of time steps required to do so can be prohibitive, especially as the desired level of accuracy becomes increas-

ingly stringent. For simulations which require a high degree of accuracy, the reduction in the number of time steps enabled by high-order methods outweighs their increased computational cost per time step. In contrast to linear multistep methods [30], unconditionally stable implicit Runge-Kutta methods can be derived for arbitrarily high orders of accuracy [33].

It is important to note that there are other more generalized classes of time-marching methods that enable the construction of high-order unconditionally stable time-marching methods. For example: advanced-step-point methods [21, 22, 66, 67], cyclic and composite linear multistep methods [8, 32, 45, 72], hybrid methods [36], multistep Runge-Kutta [42], and more broadly, general linear methods [14]. Consideration of these methods is left to a future paper.

Often in the construction of high-order implicit Runge-Kutta methods, several coefficients will remain undetermined after solving the desired order [15, 41, 58] and stability [13, 17, 28, 30, 34] conditions. This can also be created artificially by increasing the number of stages for a fixed order of accuracy. At lower orders, optimal selection of coefficients can be done analytically. In this case, there are typically fewer undetermined coefficients, and the expressions generated by the order conditions are fairly simple. As the order is increased, the size and complexity of the expressions grow, making analytical optimization intractable. To address this challenge, two approaches are common: 1) heuristic selection of undetermined coefficients; and 2) numerical optimization. In the latter case, several authors have used numerical optimization to improve the performance of Runge-Kutta methods applied to linear ODEs [7, 9, 49, 79, 80]. Still others have sought to minimize the violation in the full conditions one order higher than the scheme [10, 11, 63].

In this article we apply constrained numerical optimization to the construction of high-order unconditionally stable diagonally-implicit Runge-Kutta methods. The undetermined coefficients are optimized with respect to objective functions of accuracy, stability, and computational cost, and constrained by the desired linear or algebraic stability criteria. This is similar to the second optimization step in the work of Parsani *et al.* [63], extending it to implicit methods. It also broadens the study of optimized Runge-Kutta methods presented in Boom and Zingg [11] and Boom [10]. The novel Runge-Kutta methods optimized in this article are compared to a wide range of methods from the literature, in particular those discussed in the comprehensive review of diagonally-implicit Runge-Kutta methods presented by

Kennedy and Carpenter [53]. Several of the novel methods are found to have higher relative efficiency than the reference methods. The properties of the novel optimized methods are confirmed through numerical simulation of van der Pol's equation, along with an evaluation of their numerical performance through numerical simulation of vortex shedding in the laminar wake of a circular cylinder and the turbulent wake of a NACA 0012 airfoil at high angle-of-attack.

Section 2 reviews the common properties of Runge-Kutta methods forming the foundation of the optimization procedure presented in Section 3. The procedure is applied in Section 4 to the construction of novel unconditionally stable diagonally-implicit Runge-Kutta methods. The characteristics and performance of the methods are compared in Section 5 with numerical simulation. Finally, a summary of the most promising optimized schemes and conclusions are presented in Sections 6 and 7, respectively.

2. Diagonally-Implicit Runge-Kutta Methods

This section presents a brief review of diagonally-implicit Runge-Kutta methods and their associated properties discussed in subsequent sections. Given the initial-value problem (IVP)

$$\mathcal{Y}' = \mathcal{F}(\mathcal{Y}, t), \quad \mathcal{Y}(t_0) = y_0, \quad t_0 \leq t \leq t_f, \tag{1}$$

a Runge-Kutta method approximates the solution at a future point in time as

$$y^{[n+1]} = y^{[n]} + h \sum_{j=1}^{s} b_j \mathcal{F}(Y_j, t^{[n]} + h\, c_j), \tag{2}$$

where n is the time step index such that $y^{[n]} \approx \mathcal{Y}(t^{[n]})$ for $t^{[n]} = t_0 + hn$ and h is the time step size. The s internal stage approximations are

$$Y_k = y^{[n]} + h \sum_{j=1}^{s} A_{kj} \mathcal{F}(Y_j, t^{[n]} + h\, c_j) \quad \text{for } k = 1, \ldots, s. \tag{3}$$

The coefficients of a particular scheme are defined by the matrix A and the vector $\mathbf{b}$, along with the abscissa vector $\mathbf{c} = A\mathbb{1}$, where $\mathbb{1} = [1, \ldots, 1]^T$. Note that this definition can easily be extended to systems of differential equations using Kronecker products.

2.1. Forms of Implicit Runge-Kutta Methods

Diagonally-implicit. Implicit Runge-Kutta methods can be classified based on the form of their A coefficient matrix. In this article we focus on diagonally-implicit methods, characterized by a lower triangular A coefficient matrix with nonzero diagonal [5, 27, 55, 60]. In this case, the solution to each stage is independent of subsequent stage values; therefore, each stage can be solved sequentially [1].

Singly-diagonally-implicit. One approach to solving the resulting nonlinear system of equations at each stage is to use a Newton-type process. The residual equation for an intermediate stage (3) can be written as

$$\mathcal{R}_k = \frac{Y_k - y^{[n]}}{h} - \sum_{j=1}^{k} A_{kj} \mathcal{F}(Y_j, t^{[n]} + h\, c_j) = 0. \tag{4}$$

The Jacobian with respect to the given internal stage approximation Y_k is then

$$\mathcal{A}_k = \frac{\partial \mathcal{R}_k}{\partial Y_k} = \frac{1}{h} I - A_{kk} \frac{\partial \mathcal{F}(Y_k, t^{[n]} + h\, c_k)}{\partial Y_k}, \tag{5}$$

where I is the identity matrix. If the second term in the Jacobian $\frac{\partial \mathcal{F}(Y_k, t^{[n]} + h\, c_k)}{\partial Y_k}$ varies slowly with respect to the step size h, then it can be approximated as constant over one or more time steps. Furthermore, if the diagonal elements of the A coefficient matrix are equal, then the Jacobian can be held constant as well. Thus operations on the Jacobian, for example an LU decomposition or the construction of a preconditioner, can be reused for each stage throughout those time steps without any significant reduction in convergence [27, 60]. Methods with a constant diagonal coefficient are often called singly-diagonally implicit [44].

Explicit first stage. An explicit first stage with $c_1 = 0$ is required for a diagonally-implicit Runge-Kutta method to obtain stage order two (See *e.g.* [3]). This has a particular impact on the convergence rate of solutions to very stiff ordinary differential equations or differential algebraic equations [43, 44]. If stage order two is not required, an explicit first stage can also provide additional free coefficients for a fixed number of implicit stages to satisfy higher-order conditions or minimize truncation error. The addition of an explicit first stage is relatively inexpensive, especially when compared to the computation of the other implicit stages. Furthermore, if the method

satisfies $c_s = 1$ and $\mathbf{b}^T = A_{s,:}$, the conditions for stiff-accuracy discuss further below, the explicit first stage need only be computed for the first time step [48]. An alternative way of looking at the inclusion of an explicit first stage is as a two-value multistage-Nordsieck method [76]. The explicit first stage is equivalent to passing the derivative of the solution from the previous time step forward in addition to the solution itself.

Stiff accuracy. One of the issues with implicit time-marching methods is a phenomenon known as order reduction in which the convergence for stiff IVPs is lower than classical theory predicts. A well-known investigation of this phenomenon was done by Prothero and Robinson [65] in which they introduce the concept of stiff accuracy. While stiff accuracy does not eliminate order reduction, it influences both the stability and rate of convergence of numerical solutions to linear IVPs with stiff source terms. This concept has also become important in the derivation of convergence rates for nonlinear differential algebraic and singular perturbation problems, for example van der Pol's equation (See *e.g.* [44] and Section 5). The algebraic conditions for stiff accuracy are

$$c_s = 1 \quad \text{and} \quad \mathbf{b}^T = A_{s,:}, \tag{6}$$

implying that $y^{[n+1]} = Y_s$.

2.2. Order Conditions

Full order conditions. A Runge-Kutta method is said to be of order p if the local error is

$$e = y^{[n]} - \mathcal{Y}(t_0 + n\,h) = \mathcal{O}(h^{p+1}). \tag{7}$$

Using Butcher series, it can be shown that a Runge-Kutta method will be of order p if [15, 41, 58]

$$\mathtt{O}(\mathtt{t}) = 1 - \rho(\mathtt{t})\mathbf{b}^T \prod_{k=1}^{m} \mathtt{Y}(\mathtt{t}_k) = 0, \quad \forall \mathtt{t} \text{ such that } 1 \le \rho(\mathtt{t}) \le p, \tag{8}$$

where $\mathtt{t} = [\mathtt{t}_1, \ldots, \mathtt{t}_m]$ is a rooted tree of order $\rho(\mathtt{t})$ with m subtrees joined by a single branch to the root, the product $\prod$ is computed element wise, and

$$\mathtt{Y}(\mathtt{t}) = \begin{cases} \mathbb{1} & , \text{ if } \rho(\mathtt{t}) = 0 \\ \rho(\mathtt{t}) A \prod_{k=1}^{m} \mathtt{Y}(\mathtt{t}_k) & , \text{ otherwise} \end{cases}. \tag{9}$$

Note that the empty tree $\emptyset$ with $\rho(\emptyset) = 0$ is assumed to be a subtree of any terminal vertex in a given tree. The full order conditions are not necessarily easy to solve, but form a necessary and sufficient set of conditions for order p.

In the literature these order conditions are often scaled by a factor of $1/\left(\rho(\mathtt{t})\prod_{k=1}^{m}\rho(\mathtt{t}_k)\right)$. This is obtained by simplifying the comparison of terms in the Taylor series expansions of the exact and numerical solutions. The form presented above is derived by comparing the derivatives of the exact and numerical solutions, which is more common when considering the broader class of general linear methods. Either scaling will lead to methods of a prescribed order; however, the magnitude of the violation in the conditions of order greater than p will differ. The ultimate goal of the numerical tool presented in this article is to consider the broader class of general linear methods, thus the choice of scaling presented above. Furthermore, the numerical results presented in Section 5 support the use of this scaling to predict relative efficiency.

Simplifying order conditions. To ease the construction of higher-order Runge-Kutta methods, a number of simplifying conditions were derived by Butcher [16]. These conditions are sufficient, but not necessary for order p. In this article, the full order conditions are used in the development and optimization of new Runge-Kutta methods with the goal of using the additional degrees of freedom to improve efficiency and robustness. However, the degree to which each method satisfies the simplifying conditions is presented for reference. Furthermore, these conditions are implemented in the optimization procedure described below and could be used in the future to initialize the coefficients, or to form part of an objective function. The simplifying order conditions are summarized in the following theorem from Butcher [16]:

Theorem 1. *If the coefficients A, $\mathbf{b}$, and $\mathbf{c}$ of an Runge-Kutta method satisfy the following conditions:*

$$\mathbf{B}(p): \quad \mathbf{b}^T\mathbf{c}^{j-1} = \frac{1}{j}, \quad j \in [1,p], \tag{10}$$

$$\mathbf{C}(q): \quad A\mathbf{c}^{j-1} = \frac{\mathbf{c}^j}{j}, \quad j \in [1,q], \tag{11}$$

$$\mathbf{D}(\xi): \quad j\mathbf{b}^T C^{j-1} A = \mathbf{b}^T(I - C^j), \quad j \in [1,\xi], \tag{12}$$

with $p \leq 2q + 2$ and $p \leq q + \xi + 1$, where C is a diagonal matrix formed by the entries of $\mathbf{c}$*, then the Runge-Kutta method will be of order p.*

The first condition $\mathbf{B}(p)$ (10) is the requirement that $\mathbf{b}$ be a quadrature rule of order p. This corresponds to the full order conditions for "bushy" trees: trees for which all subtrees are of order 1. The second condition $\mathbf{C}(q)$ (11) is the requirement that the Runge-Kutta method have stage order q. Stage order is the accuracy to which the stage values $\mathbf{Y}$ approximate the solution $\mathcal{Y}(t^{[n]}\mathbb{1} + \mathbf{c})$ [16]. It is not uncommon for Runge-Kutta methods to have mixed stage order, meaning that each stage may satisfy a different number of stage-order conditions. In this case, the stage order is taken as the minimum across all stages. Finally, the second and third simplifying conditions, $\mathbf{C}(q)$ (11) and $\mathbf{D}(\xi)$ (12), imply the equivalence of certain order conditions based on the order and organization of subtrees, thus reducing the total number of conditions to be satisfied.

2.3. Local Truncation Error

For linear differential equations of the form

$$\mathcal{Y}'(t) = \lambda \mathcal{Y}(t), \quad \mathcal{Y}(t_0) = y_0, \tag{13}$$

where $\mathcal{Y}(t) \in \mathbb{C}$ and λ is a complex constant, the exact solution can be written as

$$\mathcal{Y}(t) = y_0 e^{\lambda t} = y_0 \sum_{j=0}^{\infty} \frac{(\lambda t)^j}{j!}. \tag{14}$$

Applying a Runge-Kutta method, the internal stage approximations (3) are solved by the matrix equation

$$\mathbf{Y} = (I - zA)^{-1}\mathbb{1} y^{[n]}, \tag{15}$$

where $z = \lambda h$. Substituting this result into the solution update (2) and simplifying gives

$$y^{[n+1]} = [1 + zb^T(I - zA)^{-1}\mathbb{1}]y^{[n]} = M_{\mathrm{A}}(z)y^{[n]}, \tag{16}$$

where $M_A(z)$ is called the iteration or linear stability polynomial. Expanding using Taylor series and comparing to the exact solution, the local truncation

error (LTE) coefficient $\mathcal{C} \in \mathbb{R}$ is derived:

$$M_{\mathrm{A}}(z) = 1 + zb^T(I - zA)^{-1}\mathbb{1} = \sum_{j=0}^{p} \frac{z^j}{j!} + \mathcal{O}(h^{p+1})$$
$$= e^z + \mathcal{C}\frac{z^{p+1}}{(p+1)!} + \mathcal{O}(h^{p+2}). \tag{17}$$

When p is odd the leading error term is dissipative; when p is even it is dispersive. While this is strictly linear error analysis, these properties can be important for some nonlinear problems as well, such as aeroacoustics. In this case, accurate propagation of acoustic waves to the farfield is critical, and becomes essentially linear away from the source.

2.4. Stability

A-stability. Consider the linear IVP (13) once more. It is well known that this equation is inherently stable for $\mathrm{Re}(\lambda) \leq 0$ (*e.g.* [30, 57]). In the discrete case, unconditional stability of the numerical solution to the linear IVP (13) is defined as A-stability [30]. A Runge-Kutta method is called A-stable if the polynomial $M_{\mathrm{A}}(z) \leq 1$ for all $z = \lambda h$ in the left-half complex plane, including the imaginary axis.

L-stability. While A-stability is a desirable property, parasitic components of the solution may be damped very slowly. This motivates an extension to the definition of A-stability, called L-stability [34]. A Runge-Kutta method applied to the linear IVP (13) is called L-stable if it is A-stable, and $\mathrm{Re}(\lambda) \leq 0$ implies that

$$M_{\mathrm{A}}(z) \to 0 \text{ as } |\lambda| \to \infty. \tag{18}$$

In addition to unconditional stability, this guarantees damping of stiff parasitic modes.

Internal A- and L-stability. It is possible to define similar stability properties for the internal stages of a Runge-Kutta method, called internal A- and L-stability [55]. In this case, we consider the stability polynomials

$$M_{\mathrm{A}}(z, i) = 1 + zA_{i,:}(I - zA)^{-1}\mathbb{1}, \quad \text{for } i = 1, \ldots, s, \tag{19}$$

and the definitions of internal A and L-stability are analogous to those above.

Algebraic stability. A nonlinear IVP (1) is said to be contractive if it satisfies the one-sided Lipschitz condition [31]

$$\mathrm{Re}[(\mathcal{F}(\mathcal{Y},t)-\mathcal{F}(\hat{\mathcal{Y}},t)\ ,\ \mathcal{Y}-\hat{\mathcal{Y}})] \leq \nu|\mathcal{Y}-\hat{\mathcal{Y}}|^2, \quad \forall \mathcal{Y}(t),\hat{\mathcal{Y}}(t)\in\mathbb{C}^m, \quad t\in\mathbb{R}, \tag{20}$$

with one-sided Lipschitz constant $\nu \leq 0$, where $\mathcal{Y}$ and $\hat{\mathcal{Y}}$ are two solutions defined by the initial conditions y_0 and $\hat{y}_0$, respectively. This implies that the distance between the two solutions $|\mathcal{Y}(t)-\hat{\mathcal{Y}}(t)|$ is a non-increasing function of time [44]. In the discrete case, this property is called B-stability[2]. Algebraic criteria for B-stability were derived by Burrage and Butcher [13], and given the name algebraic stability. A Runge-Kutta method is called algebraically stable if

1. $b_i \geq 0$ for $i = 1,\ldots,s$, and
2. $M_{\mathrm{BN}} = B_d A + A^T B_d - \mathbf{b}\mathbf{b}^T$ is non-negative definite,

where B_d is a diagonal matrix formed by the elements of $\mathbf{b}$.

Internal algebraic stability. The internal algebraic stability matrix for each stage is defined as

$$M_{\mathrm{BN}}(i) = A_d(i)A + A^T A_d(i) - A_{i,:}A_{i,:}^T, \tag{21}$$

where $A_d(i)$ is a diagonal matrix formed by the elements of $A_{i,:}$. As before, we desire that the internal algebraic stability matrix for each stage be non-negative definite. Internal algebraic stability also requires that the coefficients of the A coefficient matrix be non-negative.

3. Numerical Optimization of Coefficients

This section presents a numerical tool developed for the construction and optimization of implicit Runge-Kutta methods. While the focus of this article is on diagonally-implicit Runge-Kutta methods, the tool can equally be applied to fully explicit or implicit schemes as well. The tool is implemented as a custom package in Maple 18.

[2]B-stability is sometimes referred to as BN-stability when the distinction between autonomous and non-autonomous IVPs is made (Compare Definitions 2.9.2 and 2.9.3 of [50] and Definition 12.2 in [44]).

3.1. Design Variables

A number of parameters must be selected at the outset, including the number of stages s, the form of the A coefficient matrix, whether or not to enforce stiff accuracy, as well as the desired order p and stage order q of the method. These parameters are used to determine the relationship between as many coefficients as possible. Careful selection of the sequence in which the coefficients are solved can aid this process. For example, we typically solve for the abscissa values last as they appear with higher exponents in the order conditions. Solving for the abscissa values often increases the cost of the solution process since it can lead to multiple solutions. The supplementary conditions for L-stability and internal L-stability, ($\lim_{z\to\infty} M_{\mathrm{A}}(z) = 0$ and $\lim_{z\to\infty} M_{\mathrm{A}}(z, i) = 0$ for $i = 1..s$), may also be used to determine additional relationships between coefficients. The remaining undetermined coefficients are set as design variables.

3.2. Objective Function

A key metric used to evaluate the performance of a new method is efficiency. The order as well as the number of implicit stages plays a large role in the efficiency of the resulting method; however, they are set *a priori*. Therefore, to find an optimal set of coefficients, the primary objective function chosen is the L_2-principal error norm [64]:

$$\mathcal{J}_{\mathcal{E}} = \mathcal{E}(p) = \sqrt{\sum_{\forall \mathtt{t} | \rho(\mathtt{t}) = p+1} \mathtt{O}(\mathtt{t})^2}, \tag{22}$$

where $\mathtt{O}(\mathtt{t})$ are the order conditions (8). The objective function is evaluated during the optimization by first substituting the design variables into the coefficient vectors and matrices, then numerically computing the violation in $p + 1$ order conditions for the L_2-principal error norm. This is in contrast to precomputing an expression for the objective function before the optimization begins, then substituting the design variables into this expression during the optimization. The former approach minimizes the number of large complex symbolic expressions which need to be computed and stored.

One of the factors that can influence the difficulty in solving the implicit system at each stage is the relative spacing of the abscissa values. The closer the abscissa values, the better the solution at the previous stage will be as an initial guess for the current stage solution. To account for this in the optimization, the objective function discussed above can be multiplied

by some function of the relative spacing in the abscissa. In this article we consider the following modified objective function:

$$\mathcal{J}_{c\mathcal{E}} = \mathcal{P}_c \sqrt{\sum_{\forall \mathtt{t}|\rho(\mathtt{t})=p+1} \mathtt{O}(\mathtt{t})^2}, \tag{23}$$

where the premultiplying function is

$$\mathcal{P}_c = \sqrt{\left([\mathbf{c}^T\ 1] - [0\ \mathbf{c}^T]\right)\left([\mathbf{c}^T\ 1] - [0\ \mathbf{c}^T]\right)^T}. \tag{24}$$

In this article the linear stability of the method is constrained; however, the algebraic and internal stability of the method can also have an influence on the computational efficiency and stability of the method. To address this potential shortcoming, the objective function can be modified by adding the violation in these stability criteria. The resulting modified objective function can have one of the following forms:

$$\begin{aligned} \mathcal{J}_{c\mathcal{E}s1} &= \sqrt{\textstyle\sum_{\forall \mathtt{t}|\rho(\mathtt{t})=p+1} \mathtt{O}(\mathtt{t})^2} + \mathcal{P}_{s1}, \\ \mathcal{J}_{c\mathcal{E}s2} &= \sqrt{\textstyle\sum_{\forall \mathtt{t}|\rho(\mathtt{t})=p+1} \mathtt{O}(\mathtt{t})^2} + \mathcal{P}_{s1} + \mathcal{P}_{s2}, \text{ or} \\ \mathcal{J}_{c\mathcal{E}s2} &= \sqrt{\textstyle\sum_{\forall \mathtt{t}|\rho(\mathtt{t})=p+1} \mathtt{O}(\mathtt{t})^2} + \mathcal{P}_{s3}, \end{aligned} \tag{25}$$

where

$$\begin{aligned} \mathcal{P}_{s1} &= -\textstyle\sum_{\forall \lambda} \min\left(0, \mathbb{R}(\lambda_{M_{BN}})\right), \\ \mathcal{P}_{s2} &= -\textstyle\sum_i \sum_{\forall \lambda} \min\left(0, \mathbb{R}(\lambda_{M_{BN}(i)})\right), \\ \mathcal{P}_{s3} &= \textstyle\sum_i M_A(\infty, i), \end{aligned} \tag{26}$$

and $\mathbb{R}$ denotes the real component. The modified objective functions can also be combined with the efficiency metric as follows

$$\begin{aligned} \mathcal{J}_{c\mathcal{E}s1} &= \mathcal{P}_c \left(\sqrt{\textstyle\sum_{\forall \mathtt{t}|\rho(\mathtt{t})=p+1} \mathtt{O}(\mathtt{t})^2} + \mathcal{P}_{s1}\right), \\ \mathcal{J}_{c\mathcal{E}s2} &= \mathcal{P}_c \left(\sqrt{\textstyle\sum_{\forall \mathtt{t}|\rho(\mathtt{t})=p+1} \mathtt{O}(\mathtt{t})^2} + \mathcal{P}_{s1} + \mathcal{P}_{s2}\right), \text{ and} \\ \mathcal{J}_{c\mathcal{E}s2} &= \mathcal{P}_c \left(\sqrt{\textstyle\sum_{\forall \mathtt{t}|\rho(\mathtt{t})=p+1} \mathtt{O}(\mathtt{t})^2} + \mathcal{P}_{s3}\right). \end{aligned} \tag{27}$$

Alternative objective functions can easily be implemented based on the characteristics of the IVPs for which the methods are being constructed. For

example the objective function could be the coefficient of the local truncation error, or some function of the dissipative and dispersive properties of the method [49].

The gradient of the objective functions is computed using the complex step method [74] with a step size of 10^{-20}. If the coefficients develop a small imaginary component due to inexact arithmetic, a finite-difference gradient is also implemented. A typical finite-difference step size is 10^{-10}. The step sizes are chosen to maximize accuracy, and in the case of finite-differences, to minimize round-off error (See *e.g.* [40]). Note also that the code runs with double precision.

3.3. Constraints

Stability criteria, as well as bounds on the coefficients, are enforced through the use of linear and nonlinear constraints. Similar to the objective function, the constraints are evaluated during the optimization after first substituting the design variables into the coefficient vectors and matrices. Many of these constraints involve absolute values; therefore the gradient is computed with finite differences and a step size of 10^{-10}.

A-stability. The constraint for A-stability is implemented using the stability contour, which is obtained by solving $M_A(z) = e^{\mathtt{i}\theta}$ for the associated complex z-coordinate, where $\mathtt{i} = \sqrt{-1}$ and $\theta \in [0, 2\pi)$. Observe that the magnitude of $e^{\mathtt{i}\theta}$ is unity, thus defining the boundary between the stable and unstable regions. A-stability requires that the stability contour be in the right-half complex z-plane, including the imaginary axis. Given that the stability contour is symmetric about the real z-axis, the the numerical tool requires that only the domain $\theta \in [0, \pi)$ be discretized. Typically 40 non-equidistant points are used clustered around the intersection of the contour and the origin. The real components of the z-coordinates are then solved and constrained to be greater than or equal to zero. An additional constraint, $M_{\mathrm{A}}(z = -1) \leq 1$, is used to ensure that it is the stable region of the contour which lies in the left-half complex plane.

L-stability. The L-stability condition, $M_A(z) \to 0$ as $z \to \infty$, can be solve before the optimization to determine the relationship between additional coefficients. This is the preferred approach. Alternatively, the condition can be enforced as a constraint during the optimization.

Algebraic stability. The constraints for algebraic-stability are implemented with the simple inequalities:

$$-\lambda_{M_{BN},i} \leq 0, \quad \text{and} \quad -b_i \leq 0 \quad \text{for} \quad i = 1, \ldots, s, \tag{28}$$

where $\lambda_{M_{BN},i}$ are the eigenvalues of M_{BN}.

Internal stability. Internal linear and algebraic stability are enforced in much the same way as the standard stability constraints using the modified definitions presented in Section 2.4.

Coefficient Bounds. The design variables are bound during the optimization to ensure a well-conditioned method:

$$\pm(A_{ij} - \Gamma) \leq 0, \quad \pm(b_i - \Gamma) \leq 0, \quad \text{and} \quad \pm(c_i - \Gamma) \leq 0 \quad \text{for} \quad i, j = 1, \ldots, s. \tag{29}$$

where Γ is the bounding value. A typical bound is 100, though it is rarely active at convergence. The bounds can vary between individual coefficients. For example, the abscissa values can be constrained to be within a time step, in the domain $[0, 1]$.

3.4. Optimization Strategy

The sequential quadratic programming (SQP) method of Maple's nonlinear optimization construct is used to optimize the design variables subject to the nonlinear constraints. The design space is often multi-modal, having several local minima; therefore a multi-start procedure is used [24]. Initial values for the design variables are generated with a Sobol sequence [24, 51, 73] in a predetermined range. Typically 200 initial solutions are generated in the range $[-1, 1]$. The various initial conditions are then distributed and optimized in parallel. It is important to note that since the design space can be multi-modal, the results of the optimizations can only be interpreted as local minima, rather than a global minimum. Increasing the number of initial conditions increases the likelihood of finding the global minimum.

4. Optimized Runge-Kutta Methods

This section presents a number of novel methods derived using the optimization procedure described in Section 3. The focus is on L-stable singly-diagonally-implicit methods, including those with an explicit first stage.

Some A-stable and algebraically stable methods are also discussed. The novel methods are compared to a number of methods from the literature. These reference methods were largely taken out of books on time-marching methods and review articles, along with those found in the course of study. It is impossible to do an exhaustive search given the size of the literature on Runge-Kutta methods; however, over 130 new and reference methods were considered in this study (See Appendix A). In the discussion below, we only present the properties of the most relevant schemes. In this article we use a similar form of identification for Runge-Kutta methods as that used by Kennedy and Carpenter [53]:

$$\text{ESDIRK[p,(q}_i\text{)](s)X\_SA\_i}$$

E	Explicit first stage
S	Singly
DI	Diagonally-implicit
RK	Runge-Kutta
p	Order of the method
q_i	Stage order of the individual stages - if the stage orders are equal for all s stages, or the first $s-1$ stages for stiffly accurate methods, only the minimum stage order is shown
s	Number of stages
X	Stability property (A,L,Alg)
SA	Stiffly accurate
i	Unique identifier

4.1. Comparison of Diagonally-Implicit Time-Marching Methods

Efficiency is defined by the relationship between solution error and computational work. The L_2-principal error norm (22) defined above does not take into account the relative computational cost of each method. Therefore, to investigate the relative efficiency of diagonally-implicit Runge-Kutta methods of a particular order, the relative error norm is introduced [10]:

$$\mathcal{E}_{\text{rel}}(p) = \mathcal{E}(p) s_{\text{i}}^p, \tag{30}$$

where s_{i} is the number of implicit stages. The number of implicit stages is used here as a crude approximation of computational work. In reality there are many factors which can influence the efficiency of a scheme. The approximation given above is designed only to give a first order indication of relative efficiency; numerical simulation is required to more fully compare the schemes. This is presented in Section 5.

4.2. Low-order Methods

Low-order ($p \leq 2$) methods have been thoroughly investigated in the literature. When seeking maximal order with a given set of parameters, lower-order methods tend have few free variables to optimize. Furthermore, global minima can be found analytically due to the simplicity of the expressions resulting from the order conditions. As a result, applying numerical optimization to methods of order two or less yields known minima, along with a few new methods with similar accuracy and stability properties.

4.3. Order Three Methods

Table 1 presents the properties of the most relevant third-order methods optimized in this work, along with the details of some reference methods from the literature. The methods are organized by the number of stages, then the predicted efficiency based on the relative L_2-principal error norm. In this case, numerical optimization is able to generate methods with improved efficiency relative to the references. We also compare the trade-offs in stability properties and the spacing in the abscissae.

First, consider three-stage methods which are not stiffly accurate. The reference methods are Methods 3-8 and 3-9, both of which are algebraically stable. Optimizing solely with respect to the error norm, Methods 3-6 and 3-7 improve this metric by about 14% over the reference Method 3-8. In addition, both optimized methods improve the internal algebraic stability and are L-stable. However, the optimized methods have abscissae which extend beyond the time step, and Method 3-7 has a negative abscissa value. These properties did not affect the simulations presented in this article (See Section 5); however, there are some ODEs for which they can cause issues. The user should be aware of the requirements of the problem they are solving, and use the appropriate method. This highlights the potential danger in optimizing a method without proper awareness of the necessary constraints.

Forfeiting algebraic stability, several optimized methods are derived with significantly lower error norms: Methods 3-2 through 3-5. These methods are all L-stable and their abscissae lie within the time step. The most efficient of these schemes is Method 3-2; however, the violation in the global and internal algebraic stability properties is very large. By coincidence, this is improved by increasing the stage order of stages two and three, as seen with Method 3-3. Adding the global algebraic stability to the objective function, rather than strictly enforcing it, yields Method 3-5. The method has an error norm more than three times smaller than the reference Methods 3-8 and 3-9,

#	Method	Objective function	$p_{\text{Simp}}[p,\eta,\xi]$	$\mathcal{P}_c$	$\min(0,\mathbf{c}) \to \max(1,\mathbf{c})$	$M(\infty)$	$\max_i M(\infty,i)$	$\mathcal{P}_{s1}$	$\mathcal{P}_{s2}$	$\mathcal{E}(p)$	$\mathcal{E}(p)_{\text{rel}}$	Reference
3-1	SDIRK[3,1](2)Alg_cr	$\mathcal{J}_{\mathcal{E}}$	3 [4,1,1]	1.26	$0.00 \to 1.00$	0.73	0.00	0.00	1.24	2.27	18.17	*[27]
3-2	SDIRK[3,1](3)L_13	$\mathcal{J}_{\mathcal{E}}$	2 [3,1,0]	0.59	$0.00 \to 1.00$	0.00	0.00	40.76	20.10	0.63	16.97	*
3-3	SDIRK[3,(1,2,2)](3)L_14	$\mathcal{J}_{\mathcal{E}}$	2 [3,1,0]	0.77	$0.00 \to 1.00$	0.00	0.00	0.19	1.21	0.67	17.96	*
3-4	SDIRK[3,1](3)L_SA_al	$\mathcal{J}_{\mathcal{E}}$	2 [3,1,0]	0.59	$0.00 \to 1.00$	0.00	0.00	1.35	1.35	0.69	18.51	*[1, 70]
3-5	SDIRK[3,1](3)L_01	$\mathcal{J}_{\mathcal{E}\,s1}$	2 [3,1,0]	0.86	$0.00 \to 1.00$	0.00	0.00	0.06	0.33	0.71	19.04	*
3-6	SDIRK[3,1](3)AlgL	$\mathcal{J}_{\mathcal{E}}$	3 [3,1,1]	2.59	$0.00 \to 2.27$	0.00	0.00	0.00	1.78	1.89	51.16	*
3-7	SDIRK[3,1](3)AlgL_01	$\mathcal{J}_{\mathcal{E}}$	3 [3,1,1]	2.59	$-1.27 \to 1.00$	0.00	0.00	0.00	4.42	1.89	51.16	*
3-8	SDIRK[3,1](3)Alg_no2	N/A	2 [3,1,0]	1.45	$0.00 \to 1.00$	0.73	0.00	0.00	44.76	2.22	59.82	[61]
3-9	SDIRK[3,1](3)Alg_no	N/A	3 [3,1,1]	1.31	$0.00 \to 1.00$	0.73	0.00	0.00	1.26	2.23	60.12	[61]
3-10	SDIRK[3,(1,2,3,3)](4)L_11	$\mathcal{J}_{\mathcal{E}}$	2 [3,1,0]	0.78	$0.00 \to 1.00$	0.00	0.00	2.44	2.27	0.03	2.17	*
3-11	SDIRK[3,1](4)L_SA_5	$\mathcal{J}_{\mathcal{E}}$	2 [3,1,0]	0.51	$0.00 \to 1.00$	0.00	0.00	0.11	3.23	0.08	4.96	*
3-12	SDIRK[3,(1,2,2,3)](4)L_SA_7	$\mathcal{J}_{\mathcal{E}}$	2 [3,1,0]	0.69	$0.00 \to 1.00$	0.00	0.00	0.33	0.33	0.16	10.46	*
3-13	SDIRK[3,(1,1,2,3)](4)L_SA_ca	N/A	2 [3,1,0]	0.73	$0.00 \to 1.00$	0.00	0.00	0.51	0.51	0.18	11.75	[18]

Table 1: Summary of optimized third-order methods (denoted with the symbol *), along with some methods from the literature. Note that in some instances the numerical tool rederived a known method; hence the symbol * next to the reference.

and the violation in the algebraic stability condition is small. The latter is a vast improvement over Method 3-2 and only forfeits about 12% in the error norm. This demonstrates the potential optimization can have in developing methods with a balance of competing objectives.

If stiff-accuracy is required, the optimizer rederived Method 3-4 from the literature with similar properties to Methods 3-3 and 3-5. The optimizer also rederived the two-stage Method 3-1. This method has similar stability properties to the reference Method 3-9, but is significantly more efficient. The relative error norm of this method is comparable to Methods 3-2 through 3-5.

Lastly, consider four-stage L-stable methods. The reduction in error norm relative to the two and three-stage methods obtained by exploiting the additional free coefficients is enough to overcome the increased computational cost of the additional stage. Therefore, the lowest relative error norm of the four-stage methods is about eight times smaller than the two and three-stage methods. This highlights the potential efficiency benefit that can be obtained by increasing the number of stages for a fixed order of accuracy.

With four stages the reference scheme is the stiffly-accurate Method 3-13. Through numerical optimization, Method 3-11 is derived which is nearly 60% more efficient based on the relative error norm. This improvement is due in part to lowering the order of the third stage, though the formal stage order of both methods is the same. One potential drawback of this method is the larger violation in the internal algebraic stability conditions, though neither method is algebraically stable.A second scheme is optimized for which stages two and three both obtained stage order two, Method 3-12. This scheme improves every metric relative to the reference, including reducing the error norm by 11%. Finally, if stiff accuracy is forfeited and algebraic stability is not a significant concern, Method 3-10 has an error norm more than five times smaller than the reference Method 3-13.

The higher predicted efficiency of the optimized schemes presented above could yield a significant reduction in computational resources or solution time required to obtain a result with a prescribed level of accuracy. This demonstrates the potential benefits of numerical optimization for the selection of undetermined coefficients in the construction of Runge-Kutta methods.

4.4. Order Four Methods

Table 2 presents the properties of fourth-order methods. As with the third-order methods, the most efficient methods based on the relative error norm have a greater number of stages than strictly required. Methods 4-10

through 4-14 have two more implicit stages than Method 4-1, but have relative error norms one to two orders smaller. This again highlights the potential advantage of increasing the number of stages for a fixed order of accuracy. Applying optimization to six-stage methods with respect to the error norm alone leads to fifth-order schemes. Future work will include investigation of different objective functions which allow the optimization of fixed-order schemes with a greater number of stages than required. This may be possible by including the additional stability criteria into the objective function or by considering the L_2-principal error norm of the violation in both the $p+1$ and $p+2$ order conditions.

With both three and five stages the numerical tool presented in this article rederived known methods from the literature, indicating that they are already minima in the design space. While no reference method is found with four stages, the optimization of these methods has demonstrated some of the potential advantages in using numerical optimization to achieve different objectives. Method 4-2 is optimized for the lowest error norm $\mathcal{E}(p)$. This is achieved; however the violation in the algebraic stability conditions is large. Method 4-4 is optimized with the inclusion of the cost metric $\mathcal{P}_c$. The optimization is able to lower this metric by about 11%, without significantly affecting the error norm ($\sim 1\%$ increase). However, the violation of the algebraic stability condition rises significantly and one of the coefficients of the method is large. Repeating the optimization with a reduced bound on the coefficients, from 100 to 10, yielded Method 4-6. This further reduced the cost metric and improved the algebraic stability, though it is still large. The trade-off appears in the error norm, which rises by about 5% relative to Method 4-2.

The objective function of Method 4-8 seeks to improve the algebraic stability of the methods and is able to reduce the violation in the algebraic stability conditions by 3 to 4 orders of magnitude. This comes with a 7.7% increase in the error norm and a 45% increase in the cost metric relative to Method 4-2. Adding in the cost metric to the objective function, Method 4-7 is able to improve both the cost and error norm relative to Method 4-8. The violation in the algebraic stability conditions increases by an order of magnitude, but is still 2 to 3 orders smaller than Methods 4-2, 4-4 and 4-6. However, the violation in the internal algebraic stability conditions spikes. Methods 4-9 and 4-5 add in the violation of the internal algebraic stability conditions to the objective function. Method 4-9 has the best global and internal algebraic stability properties, but also has the highest error norm and

#	Method	Objective function	$p_{\text{Simp}}[p,\eta,\xi]$	$\mathcal{P}_c$	$\min(0,\mathbf{c})-\max(1,\mathbf{c})$	$M(\infty)$	$\max_i M(\infty,i)$	$\mathcal{P}_{s1}$	$\mathcal{P}_{s2}$	$\mathcal{E}(p)$	$\mathcal{E}(p)_{\text{rel}}$	Reference
4-1	SDIRK[4,1](3)Alg_cr	$\mathcal{J}_{\mathcal{E}}$	3 [4,1,1]	1.71	$-0.07 \to 1.07$	0.63	0.00	0.00	21.39	21.00	1700.95	*[27]
4-2	SDIRK[4,1](4)L_03	$\mathcal{J}_{\mathcal{E}}$	3 [4,1,1]	0.92	$0.00 \to 1.00$	0.00	0.00	56.63	2.04	3.38	865.47	*
4-3	SDIRK[4,(1,2,2,2)](4)L_13	$\mathcal{J}_{\mathcal{E}}$	3 [4,1,1]	0.96	$0.00 \to 1.00$	0.00	0.00	8.18	1.85	3.39	866.76	*
4-4	SDIRK[4,1](4)L_01	$\mathcal{J}_{c\mathcal{E}}$	3 [4,1,1]	0.82	$0.00 \to 1.00$	0.00	0.00	723.18	5.28	3.43	878.72	*
4-5	SDIRK[4,1](4)L_05	$\mathcal{J}_{c\mathcal{E}s2}$	3 [4,1,1]	1.19	$0.00 \to 1.00$	0.00	0.00	1.08	1.37	3.53	904.84	*
4-6	SDIRK[4,1](4)L_04	$\mathcal{J}_{c\mathcal{E}}$	3 [4,1,1]	0.81	$0.00 \to 1.00$	0.00	0.00	224.16	8.75	3.55	909.92	*
4-7	SDIRK[4,1](4)L_02	$\mathcal{J}_{c\mathcal{E}s1}$	3 [4,1,1]	1.14	$0.00 \to 1.00$	0.00	0.00	0.53	49.77	3.62	926.11	*
4-8	SDIRK[4,1](4)L_00	$\mathcal{J}_{\mathcal{E}s1}$	3 [4,1,1]	1.33	$0.00 \to 1.00$	0.00	0.00	0.03	2.91	3.64	931.61	*
4-9	SDIRK[4,1](4)L_06	$\mathcal{J}_{\mathcal{E}s2}$	3 [4,1,1]	1.65	$-0.10 \to 1.03$	0.00	0.00	0.00	1.22	3.95	1011.70	*
4-10	SDIRK[4,1](5)L_SA_ha	$\mathcal{J}_{\mathcal{E}}$	2 [4,1,0]	0.78	$0.00 \to 1.00$	0.00	0.21	112.09	112.09	0.13	83.51	*[44, 53]
4-11	ESDIRK[4,2](5)L_SA_k2c	$\mathcal{J}_{\mathcal{E}}$	3 [4,2,0]	1.36	$0.00 \to 1.15$	0.00	1.00	0.70	1.20	4.63	1184.33	*[53]
4-12	ESDIRK[4,2](6)L_SA_sk4	N/A	3 [5,2,0]	1.37	$0.00 \to 1.00$	0.00	1.44	0.26	0.67	0.14	89.77	[69]
4-13	ESDIRK[4,2](6)L_SA_k2c	N/A	3 [4,2,0]	0.88	$0.00 \to 1.04$	0.00	1.00	0.20	3.36	0.16	98.45	[53]
4-14	ESDIRK[4,2](6)L_SA_kc	N/A	3 [4,2,0]	0.66	$0.00 \to 1.00$	0.00	1.00	0.49	0.95	0.19	117.12	[52, 53]

Table 2: Summary of optimized fourth-order methods (denoted with the symbol *), along with some schemes from the literature. Note that in some instances the numerical tool rederived a known method; hence the symbol * next to the reference.

value of the cost metric and abscissa values outside of the time step. Method 4-5 seems to present a good balance of the properties considered, exchanging some computational efficiency for improved algebraic stability properties.

This study highlights the need to consider multiple factors in the optimization of a method. It also highlights the ability of the optimization to derive methods subject to competing objectives. For example, one could create a Pareto front of methods relative to different objectives by manipulating their relative importance in the objective function.

4.5. Order Five Methods

Table 3 summarizes the most significant fifth-order schemes considered in this study. The reference five-stage Method 5-3 is an A-stable non-stiffly accurate method. The last three stages of this method are second-order accurate, though the formal stage order of the method is still only first order. Through numerical optimization, a novel Method 5-1 is derived which reduces the relative error norm by over 40% and adds L-stability. This optimization made use of the abscissa constraint to ensure that all values are within the time step. The method yields an increase in the violation of algebraic stability conditions and its stages are uniformly first order. A second optimization is performed to balance the violation in the global and internal algebraic stability conditions. Method 5-2 does achieve a smaller violation in the internal algebraic stability condition, but increases the error norm, the spacing in the abscissa, and violation in the global algebraic stability condition relative to Method 5-1. Method 5-2 is still markedly better than the reference Method 5-3. The significant improvements in predicted relative efficiency obtained through numerical optimization demonstrate the benefit of the approach, in particular at higher orders of accuracy.

As mentioned above, optimizing a six-stage ESDIRK method yields a fifth-order scheme. The reference Method 5-10 yields a noticeable violation in the algebraic stability constraints, and the abscissa is not contained within the time step. However, numerical optimization was not able to derive an L-stable method with the abscissa wholly contained within the time step. If one is willing to settle for A-stability, Method 5-4 has an abscissa contained within the time step and achieves a 50% reduction in the error norm. The internal linear stability is good, and the violation of the algebraic stability conditions is of the same order of magnitude as the reference Method 5-10.

Optimizing a six-stage ESDIRK method solely with respect to the error norm can yield a reduction of up to 30% in this metric. However, the resulting

#	Method	Objective function	$p_{\text{Simp}}[p,\eta,\xi]$	$\mathcal{P}_c$	$\min(0,\mathbf{c}) - \max(1,\mathbf{c})$	$M(\infty)$	$\max_i M(\infty,i)$	$\mathcal{P}_{s1}$	$\mathcal{P}_{s2}$	$\mathcal{E}(p)$	$\mathcal{E}(p)_{\text{rel}}$	Reference
5-1	SDIRK[5,1](5)L_02	$\mathcal{J}_{\mathcal{E}}$	3 [5,1,1]	1.20	$0.00 \to 1.00$	0.00	0.00	0.26	2.62	0.73	2294.64	*
5-2	SDIRK[5,1](5)L_01	$\mathcal{J}_{c\mathcal{E}s2}$	3 [5,1,1]	1.25	$0.00 \to 1.00$	0.00	0.00	0.58	0.28	0.82	2566.39	*
5-3	SDIRK[5,(1,1,2,2,2)](5)A_co	N/A	3 [5,1,1]	1.20	$0.00 \to 1.00$	0.98	0.00	0.11	0.41	1.25	3903.99	[25]
5-4	ESDIRK[5,2](6)A_SA	$\mathcal{J}_{c\mathcal{E}}$	3 [5,2,0]	1.14	$0.00 \to 1.00$	1.00	1.02	23.02	23.02	0.46	1430.45	*
5-5	ESDIRK[5,2](6)L_SA_00	$\mathcal{J}_{\mathcal{E}}$	3 [5,2,0]	0.98	$0.00 \to 1.24$	0.00	5.28	0.50	36.33	0.61	1913.77	*
5-6	ESDIRK[5,2](6)L_SA_2	$\mathcal{J}_{\mathcal{E}}$	3 [5,2,0]	0.99	$0.00 \to 1.23$	0.00	5.22	0.89	76.46	0.61	1915.45	[10]
5-7	ESDIRK[5,2](6)L_SA_01	$\mathcal{J}_{c\mathcal{E}}$	3 [5,2,0]	0.97	$0.00 \to 1.26$	0.00	5.20	0.26	11.84	0.62	1923.01	*
5-8	ESDIRK[5,2](6)L_SA_bm	N/A	3 [5,2,0]	1.29	$0.00 \to 1.09$	0.00	1.56	0.16	0.80	0.89	2773.14	[11]
5-9	ESDIRK[5,2](6)L_SA_07	$\mathcal{J}_{c\mathcal{E}s3}$	3 [5,2,0]	1.51	$-0.07 \to 1.00$	0.00	1.00	0.78	1.83	0.89	2774.12	*
5-10	ESDIRK[5,2](6)L_SA_k2c	N/A	3 [5,2,0]	1.51	$0.00 \to 1.03$	0.00	1.00	10.23	10.23	0.92	2862.26	[53]
5-11	ESDIRK[5,2](7)L_SA_k2c	N/A	3 [5,2,0]	0.74	$0.00 \to 1.04$	0.00	1.00	0.80	131.38	0.31	2395.51	[53]
5-12	ESDIRK[5,2](7)L_SA_sk	N/A	3 [6,2,0]	1.18	$0.00 \to 1.00$	0.00	5.07	0.25	2.54	0.34	2659.86	[68]

Table 3: Summary of optimized fifth-order methods (denoted with the symbol *), along with some schemes from the literature. Note that in some instances the numerical tool rederived a known method; hence the symbol * next to the reference.

Methods 5-5 through 5-7 have abscissa values far outside of the time step and significant internal linear instability. The internal linear instability can affect the robustness and efficiency of the method for some problems (See Section 5.2.1). To overcome this result, the objective function is augmented with the violation in the linear stability property, $M_A(\infty, i) = 0$. Method 5-9 is able to reduce the error norm with respect to the reference method, maintain the violation in the internal linear stability property, and reduce the violation in the algebraic stability condition. However, it introduces a negative abscissa value, which is not ideal. This method is similar to the previously published Method 5-9, exchanging mild linear instability for positive abscissa values. If we repeat the optimization described above with the constraint that the abscissa be positive, we recover a method very similar to the reference Method 5-9.

5. Numerical Simulations

This section aims to confirm the order and relative efficiency results presented in Section 4. This is accomplished with numerical simulation of both nonstiff and stiff variants of van der Pol's equation to check the order of convergence, and both laminar flow over a circular cylinder and turbulent flow over a NACA 0012 airfoil to demonstrate relative efficiency. This section also discusses the impact of various properties on the performance of the methods applied to the computational fluid dynamics simulations.

5.1. Van der Pol's Equation

Van der Pol's equation is a second-order nonlinear ODE

$$\mathcal{Y}''(x) - \mu\left(1 - \mathcal{Y}(x)^2\right)\mathcal{Y}'(x) + \mathcal{Y}(x) = 0, \tag{31}$$

which is solved as a rescaled first-order system using $\mathcal{Z}_1(t) = \mathcal{Y}'(x)$, $\mathcal{Z}_2 = \mu\mathcal{Y}''(x)$, $t = x/\mu$, and letting $\epsilon = \mu^{-2}$:

$$\left\{ \begin{array}{c} \mathcal{Z}_1'(t) = \mathcal{Z}_2(t) \\ \epsilon\mathcal{Z}_2'(t) = \left(1 - \mathcal{Z}_1(t)^2\right)\mathcal{Z}_2(t) - \mathcal{Z}_1(t) \end{array} \right\}, \tag{32}$$

where ϵ is called the stiffness parameter. This type of problem is known as a singular perturbation problem. The initial conditions, $\mathcal{Z}_1(0) = 2$ and $\mathcal{Z}_2(0) = -\frac{2}{3} + \frac{10}{81}\epsilon - \frac{292}{2187}\epsilon^2 - \frac{1814}{19683}\epsilon^3$, are chosen to give a smooth solution [44], and the time domain is set to $t = [0, 0.5]$ to be consistent with the literature

[44, 52]. The aim is to evaluate the performance of time-marching methods for stiff and nonstiff nonlinear problems. A relatively large value of $\epsilon = 0.1$ is chosen for a nonstiff problem, and a much smaller value of $\epsilon = 10^{-5}$ for a stiff problem.

The primary results are convergence rates based on the discrete L_2 norm of the solution error at end of each time step:

$$e_{z_{1/2}} = \sqrt{\frac{\sum_{i=1}^{N} (z_{1/2,i} - z_{1/2,i,\text{ref}})^2}{N}}, \tag{33}$$

where N represents the number of time steps. The reference solution is computed with ESDIRK[4,2](6)L_SA_kc and a time step 2^{-17}. This is several orders of magnitude smaller than any step size considered in the studies below.

Table 4 summarizes the convergence rates obtained for simulations of both the nonstiff and stiff van der Pol's equation. For the nonstiff problem, all the schemes recover the design order presented in Section 3. For the stiff problem, the first-order form of van der Pol's equation (32) is a singular perturbation problem. The convergence theory for this class of stiff problem is more closely related to differential algebraic equations than ordinary differential equations. Hairer *et al.* [43] showed that the convergence of a Runge-Kutta method applied to a singular perturbation problem achieves a convergence rate of

$$z_1^{[n]} - \tilde{z}_{\text{d},1}^{[n]} = \mathcal{O}(h_N^p) + \mathcal{O}(\epsilon h_N^{q+1}), \text{ and } z_2^{[n]} - \tilde{z}_{\text{d},2}^{[n]} = \mathcal{O}(h_N^{q+1}). \tag{34}$$

Furthermore, if the method is stiffly accurate, the convergence of the second variable becomes

$$z_2^{[n]} - \tilde{z}_{\text{d},2}^{[n]} = \mathcal{O}(h_N^p) + \mathcal{O}(\epsilon h_N^q). \tag{35}$$

In the stiff case, the Runge-Kutta schemes recover the order predicted by this theory. In many cases the convergence rate of the methods transition from the higher rate of p to the lower rate of q or $q+1$ as the time step is refined. The fifth-order SDIRK[5,(1,1,2,2,2)](5)A_co achieves a higher than predicted convergence for $\tilde{z}_{\text{d},2}$; however, it does not appear to be transitioning to the lower rate of $q+1$. This may be due to the higher-order accuracy in the last three stages of the method, but is unknown. Overall, these simulations verify the ability of the numerical tool presented in Section 3 to generate methods with the prescribed accuracy properties.

#	Scheme	s	p	q	p_{z_1}/p_{z_2}	
					Nonstiff ($\epsilon = 0.1$)	Stiff ($\epsilon = 10^{-5}$)
3-1	SDIRK[3,1](2)Alg	2	3	1	3.0299 / 2.9831	2.9151 / 2.0829
3-2	SDIRK[3,1](3)L_13	3	3	1	3.0225 / 2.9612	2.8397 / 2.1808
3-3	SDIRK[3,(1,2,2)](3)L_14	3	3	1	3.0024 / 2.9706	2.7132 / 2.2593
3-4	SDIRK[3,1](3)L_SA_al	3	3	1	2.9865 / 2.9859	3.1216 / 1.2044
3-5	SDIRK[3,1](3)L_01	3	3	1	3.0071 / 2.9726	2.9769 / 2.2771
3-6	SDIRK[3,1](3)AlgL	3	3	1	2.9936 / 3.0169	3.1358 / 2.4148
3-7	SDIRK[3,1](3)AlgL_01	3	3	1	3.0289 / 2.9897	2.1769 / 2.3428
3-8	SDIRK[3,1](3)Alg_no2	3	3	1	2.9813 / 2.9741	2.9489 / 2.0074
3-9	SDIRK[3,1](3)Alg_no	3	3	1	3.0295 / 2.9805	2.9204 / 2.0843
3-10	SDIRK[3,(1,2,3,3)](4)L_11	4	3	1	2.9356 / 3.3090	3.4860 / 2.0145
3-11	SDIRK[3,1](4)L_SA_5	4	3	1	2.9961 / 3.0310	3.0215 / 1.0566
3-12	SDIRK[3,(1,2,2,3)](4)L_SA_7	4	3	1	3.0034 / 2.9934	3.0125 / 0.9820
3-13	SDIRK[3,(1,1,2,3)](4)L_SA_ca	4	3	1	3.0013 / 2.9419	3.0085 / 1.1692
4-1	SDIRK[4,1](3)Alg_cr	3	4	1	3.8801 / 3.9313	2.0213 / 2.0097
4-2	SDIRK[4,1](4)L_03	4	4	1	3.9392 / 3.9551	2.0378 / 2.2780
4-3	SDIRK[4,(1,2,2,2)](4)L_13	4	4	1	3.9416 / 3.9560	2.0406 / 2.2934
4-4	SDIRK[4,1](4)L_01	4	4	1	3.8531 / 4.0139	1.8324 / 2.2087
4-5	SDIRK[4,1](4)L_05	4	4	1	3.9407 / 3.9610	2.0552 / 2.3390
4-6	SDIRK[4,1](4)L_04	4	4	1	3.8491 / 3.9186	2.2741 / 2.1995
4-7	SDIRK[4,1](4)L_02	4	4	1	3.9437 / 3.9588	2.0163 / 2.3243
4-8	SDIRK[4,1](4)L_00	4	4	1	3.9414 / 3.9622	2.0528 / 2.3426
4-9	SDIRK[4,1](4)L_06	4	4	1	3.9439 / 3.9630	2.0451 / 2.3635
4-10	SDIRK[4,1](5)L_SA_ha	5	4	1	3.9476 / 3.9623	2.4379 / 2.2324
4-11	ESDIRK[4,2](5)L_SA_k2c	5	4	2	3.8764 / 3.9397	4.0611 / 2.0462
4-12	ESDIRK[4,2](6)L_SA_sk4	6	4	2	3.9479 / 3.9776	4.2808 / 1.9356
4-15	ESDIRK[4,2](6)L_SA_kc	6	4	2	4.0178 / 4.0110	4.0511 / 2.0029
5-1	SDIRK[5,1](5)L_02	5	5	1	4.8517 / 5.0190	5.0017 / 2.0686
5-2	SDIRK[5,1](5)L_01	5	5	1	4.8744 / 4.9306	5.3009 / 2.0647
5-3	SDIRK[5,(1,1,2,2,2)](5)A_co	5	5	1	4.6262 / 4.7871	5.3756 / 2.8450
5-4	ESDIRK[5,2](6)A_SA	6	5	2	4.8415 / 4.8634	5.2847 / 2.0224
5-5	ESDIRK[5,2](6)L_SA_00	6	5	2	4.8224 / 4.8740	4.7249 / 2.0871
5-6	ESDIRK[5,2](6)L_SA_2	6	5	2	4.8197 / 4.8742	5.1336 / 2.0419
5-7	ESDIRK[5,2](6)L_SA_01	6	5	2	4.8280 / 4.8742	4.6414 / 2.0849
5-8	ESDIRK[5,2](6)L_SA_bm	6	5	2	4.8390 / 4.9100	5.2840 / 2.0796
5-9	ESDIRK[5,2](6)L_SA_07	6	5	2	4.8391 / 4.9098	5.2687 / 2.2270
5-10	ESDIRK[5,2](6)L_SA_k2c	6	5	2	4.9040 / 4.9397	5.2444 / 2.0863
5-11	ESDIRK[5,2](7)L_SA_k2c	7	5	2	4.9327 / 4.9390	4.9353 / 2.1706
5-12	ESDIRK[5,2](7)L_SA_sk	7	5	2	4.9293 / 4.9247	5.6822 / 2.2411

Table 4: **Van der Pol's equation** ($\epsilon = 0.1$ and $\epsilon = 10^{-5}$)**:** Convergence rates $p_{z_{1/2}}$ of the solution error $e_{z_{1/2}}$. The number of stages is given by s, q is the stage order, and p is the order of the scheme. The convergence rates were computed using a line of best fit through the 3 finest grid levels computed before round-off error.

5.2. Navier-Stokes and Reynolds-Averaged Navier-Stokes Equations

The flow solver chosen for this work is called Diablo and was developed at the University of Toronto Institute for Aerospace Studies. It is capable of solving both the Navier-Stokes and Reynolds-averaged Navier-Stokes equations with the negative variant of the one-equation Spalart-Allmaras turbulence model [4]. The governing equations are discretized in space by high-order classical finite-difference summation-by-parts (FD-SBP) operators and solved on structured multi-block grids. The viscous fluxes are computed with a non-compact formulation of the classical FD-SBP second-derivative operators, *i.e.* application of the first derivative twice. Simultaneous approximation terms (SATs) are used to enforce block-interface coupling and boundary conditions, while matrix artificial dissipation compatible with the FD-SBP-SAT discretization is used to maintain numerical stability. An inexact Newton-Krylov algorithm is used to drive the nonlinear residual equations to zero. The convergence of Newton's method is accelerated through the use of restricted preconditioner updates, and relative tolerance nonlinear subiteration termination, discussed further below. Finally, the linear system is solved with FGMRES and a parallel approximate-Schur preconditioner. All computations were performed on the General Purpose Cluster (GPC) at the SciNet HPC Consortium part of Compute Canada. For more information on the Diablo flow solver refer to [46, 62].

Delayed preconditioner updates. For singly-diagonally-implicit time-marching methods the temporal component of the Jacobian is constant and is often significantly larger than the change in the spatial Jacobian over a stage or an entire time step. Therefore, it is possible to freeze the preconditioner over a stage or time step without a significant impact on the convergence of the system. This reduces CPU time and thus increases the efficiency of the solution algorithm. Current results were obtained by freezing the preconditioner over each time step.

Termination of nonlinear iterations. The temporal integration has a certain level of truncation error associated with it. The convergence of the residual equations can, therefore, be terminated when the residual is less than this error. This reduces computational cost and is done without any loss in global accuracy. In this work, termination is based on a preset reduction from the initial residual value. The necessary relative tolerance is fairly step size independent since a reduction in step size will result in a better initial

iterate and therefore a lower initial residual. For the comparative simulations presented below, a conservative relative tolerance of 10^{-8} was used, requiring between 4 and 8 Newton iterations per stage.

Error. A reference solution is obtained for each of the simulations presented below. The ESDIRK[5,2](6)L_SA_k2c method with a time step size of $\Delta t = 2^{-7}$ is used for the laminar and turbulent simulations. The use of a reference case, computed on the same grid, eliminates the influence of the spatial discretization error, thus isolating the temporal error. The error is computed as the root-mean-square of the difference in lift and drag coefficients (C_L and C_D):

$$e_{C_{L/D}} = \sqrt{\frac{\sum_{i=1}^{N}(C_{L/D,i} - C_{L/D,i,\text{ref}})^2}{N}}, \tag{36}$$

where N is the number of time steps.

5.2.1. Laminar Flow Around a Circular Cylinder

In this section, two-dimensional laminar flow around a circular cylinder is simulated at Reynolds number 1200 and Mach number 0.3. Under these conditions, unsteady vortex shedding is observed in the wake of the cylinder causing regular smooth variations in lift and drag. This behaviour makes it ideal for comparing the relative accuracy and efficiency of time-marching methods [19, 75]. The results presented are obtained with a fourth-order spatial discretization on a 28,000 cell grid: 141 nodes in the offwall direction, and 200 in the circumferential direction. The offwall spacing at the surface is 0.005 times the diameter of the cylinder, and 0.5 diameters at the outer boundary, which is located 20 diameters from the surface of the cylinder. The maximum aspect ratio is about 5.3 near the leading edge. This decrease toward the leeward side of the cylinder due to the increased circumferentially resolution to capture the details of the wake. Grids are decomposed into 16 blocks and the solution is computed in parallel with a one-to-one block to processor ratio. Figure 1 shows the grid.

The temporal accuracy and efficiency of the Runge-Kutta methods are evaluated in a temporal convergence study with time step sizes from $\Delta t = 2^{-7}$ to $\Delta t = 2^3$. The simulations are run for 40 non-dimensional time units, equal to about 2.9 vortex shedding cycles. Table 5 presents the results of the simulations, including convergence rates computed with a line of best fit through the simulations with the three smallest time step before roundoff

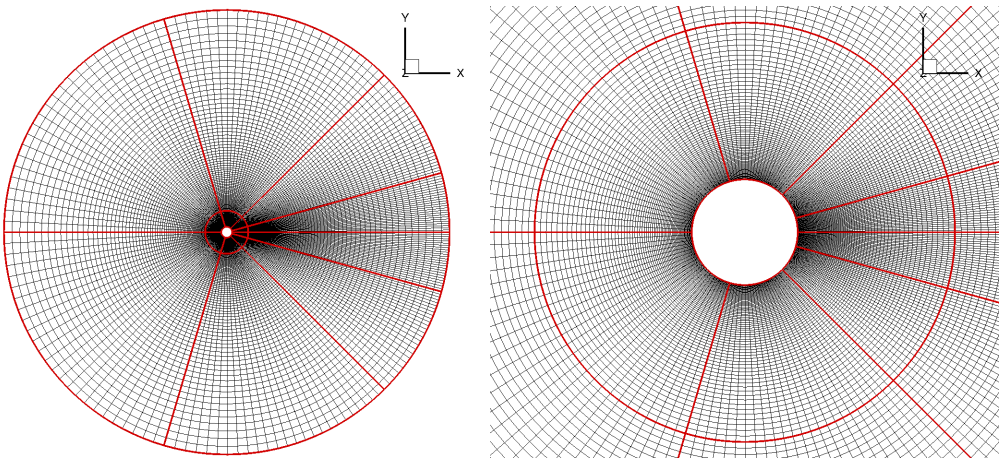

Figure 1: **Laminar Cylinder:** Computational grid with block boundaries highlighted in red

error and the total number of linear iteration required to obtain a prescribed error. Individual values are shown for lift and drag, and the number of linear iterations is interpolated for a prescribed error of $e_{C_{L/D}} = 10^{-6}$. Also shown are the average number of linear iterations per implicit stage computed from the simulation with the three smallest time step before roundoff error. The methods are organized based on order, followed by the number of stages, then the predicted relative efficiency from Section 4.

Table 5 shows that the methods of orders three and four all roughly obtain their prescribed rate of convergence. This further supports the results from simulation of van der Pol's equation. In contrast, the fifth-order methods often exhibit higher than expect convergence rates. This may be due to the methods not yet being in their asymptotic region of convergence before hitting round-off error for this problem.

One of the biggest influences on the relative efficiency of the methods applied to this problem is the local truncation error (LTE) coefficient shown in Table 6, rather than the error norm discussed to this point. This accounts for the greatest variation in the results relative to the predictions made in Section 4. It also demonstrates the need to be aware of which properties influence the efficiency of individual problems, and to optimize for these properties.

Another property which has a significant impact of the results is internal linear stability. Consider Methods 5-5 through 5-7, which have very poor in-

#	Method	C_L p	C_L Lin its ($e = 10^{-6}$)	C_D p	C_D Lin its ($e = 10^{-6}$)	Avg. Lin its/s_i
3-2	SDIRK[3,1](3)L_13	3.07	5.86e5	3.08	4.30e5	2.27e1
3-3	SDIRK[3,(1,2,2)](3)L_14	3.07	5.85e5	3.08	4.33e5	2.64e1
3-4	SDIRK[3,1](3)L_SA_al	3.07	5.86e5	3.08	4.31e5	2.27e1
3-5	SDIRK[3,1](3)L_01	3.01	5.85e5	3.02	4.33e5	2.27e1
3-6	SDIRK[3,1](3)AlgL	3.10	5.85e5	3.10	4.31e5	2.27e1
3-7	SDIRK[3,1](3)AlgL_01	3.11	5.85e5	3.12	4.28e5	2.27e1
3-8	SDIRK[3,1](3)Alg_no2	3.09	9.29e5	3.10	6.91e5	2.05e1
3-9	SDIRK[3,1](3)Alg_no	3.09	9.29e5	3.10	6.91e5	2.73e1
3-10	SDIRK[3,(1,2,3,3)](4)L_11	3.50	2.52e5	3.59	1.91e5	2.77e1
3-11	SDIRK[3,1](4)L_SA_5	3.40	2.61e5	3.43	2.00e5	2.29e1
3-12	SDIRK[3,(1,2,2,3)](4)L_SA_7	3.42	2.56e5	3.42	1.96e5	2.29e1
3-13	SDIRK[3,(1,1,2,3)](4)L_SA_ca	3.09	4.12e5	3.10	3.10e5	1.88e1
4-2	SDIRK[4,1](4)L_03	4.01	4.68e5	4.11	3.74e5	2.47e1
4-3	SDIRK[4,(1,2,2,2)](4)L_13	4.01	4.68e5	4.10	3.74e5	2.47e1
4-4	SDIRK[4,1](4)L_01	4.01	4.28e5	4.10	3.47e5	2.29e1
4-5	SDIRK[4,1](4)L_05	4.00	4.67e5	4.04	3.73e5	2.47e1
4-6	SDIRK[4,1](4)L_04	4.01	4.70e5	4.13	3.75e5	2.44e1
4-7	SDIRK[4,1](4)L_02	4.02	4.70e5	4.10	3.75e5	2.47e1
4-8	SDIRK[4,1](4)L_00	3.99	4.66e5	4.03	3.72e5	2.47e1
4-9	SDIRK[4,1](4)L_06	3.99	4.66e5	3.97	3.72e5	2.47e1
5-1	SDIRK[5,1](5)L_02	6.20	1.51e5	4.76	1.73e5	2.81e1
5-2	SDIRK[5,1](5)L_01	5.85	1.84e5	4.67	2.12e5	2.98e1
5-3	SDIRK[5,(1,1,2,2,2)](5)A_co	6.49	2.25e5	5.87	1.94e5	2.68e1
5-4	ESDIRK[5,2](6)A_SA	7.12	1.48e5	6.72	1.31e5	3.57e1
5-5	ESDIRK[5,2](6)L_SA_00	5.64	1.92e5	5.57	1.64e5	4.23e1
5-6	ESDIRK[5,2](6)L_SA_2	5.64	1.92e5	5.57	1.64e5	3.68e1
5-7	ESDIRK[5,2](6)L_SA_01	5.64	1.92e5	5.57	1.64e5	3.69e1
5-8	ESDIRK[5,2](6)L_SA_bm	6.63	1.74e5	6.41	1.53e5	3.68e1
5-9	ESDIRK[5,2](6)L_SA_07	6.79	1.72e5	6.51	1.52e5	3.68e1
5-10	ESDIRK[5,2](6)L_SA_k2c	6.81	1.72e5	6.53	1.52e5	3.68e1

Table 5: **Laminar flow over a cylinder:** Convergence rates are presented, along with the number of linear iterations required to obtain an error of 10^{-6}. These values are computed for both lift and drag. The average number of linear iterations required per implicit stage s_i is also reported.

#	Method	$\mathcal{C}$
3-2	SDIRK[3,1](3)L_13	2.59e-2
3-3	SDIRK[3,(1,2,2)](3)L_14	2.59e-2
3-4	SDIRK[3,1](3)L_SA_sk2	2.59e-2
3-5	SDIRK[3,1](3)L_01	2.59e-2
3-6	SDIRK[3,1](3)AlgL	2.59e-2
3-7	SDIRK[3,1](3)AlgL_00	2.59e-2
3-8	SDIRK[3,1](3)Alg_no2	8.80e-2
3-9	SDIRK[3,1](3)Alg_no	8.80e-2
3-10	SDIRK[3,(1,2,3,3)](4)L_11	3.79e-4
3-11	SDIRK[3,1](4)L_SA_5	3.79e-4
3-12	SDIRK[3,(1,2,2,3)](4)L_SA_7	3.79e-4
3-13	SDIRK[3,(1,1,2,3)](4)L_SA_ca	3.91e-3
4-2	SDIRK[4,1](4)L_03	2.73e-2
4-3	SDIRK[4,(1,2,2,2)](4)L_13	2.73e-2
4-4	SDIRK[4,1](4)L_01	2.73e-2
4-5	SDIRK[4,1](4)L_05	2.73e-2
4-6	SDIRK[4,1](4)L_04	2.73e-2
4-7	SDIRK[4,1](4)L_02	2.73e-2
4-8	SDIRK[4,1](4)L_07	2.73e-2
4-9	SDIRK[4,1](4)L_06	2.73e-2
5-1	SDIRK[5,1](5)L_02	5.30e-4
5-2	SDIRK[5,1](5)L_01	5.30e-4
5-3	SDIRK[5,(1,1,2,2,2)](5)A_co	1.39e-3
5-4	ESDIRK[5,2](6)A_SA	2.08e-4
5-5	ESDIRK[5,2](6)L_SA_00	5.30e-4
5-6	ESDIRK[5,2](6)L_SA_2	5.30e-4
5-7	ESDIRK[5,2](6)L_SA_01	5.30e-4
5-8	ESDIRK[5,2](6)L_SA	5.30e-4
5-9	ESDIRK[5,2](6)L_SA_07	5.30e-4
5-10	ESDIRK[5,2](6)L_SA_k2c	5.30e-4

Table 6: **Laminar flow over a cylinder:** Local truncation error coefficient for methods applied to this problem.

ternal linear stability relative to other ESDIRK[5,2] methods. Despite lower relative error norms and equal LTE coefficients, the number of linear iterations required to obtain the prescribed level of accuracy by these methods is greater than comparable methods. The only other property of these methods which is significantly different is the violation in the algebraic stability condition. However, a similar increase in linear iterations is not observed when using other methods with a large violation in the algebraic stability conditions.

As alluded to above, both global and internal algebraic stability seem to play a negligible role in the performance of the methods on this problem. The maximum feasible time step size, the iterative performance of the solver, the order, and relative efficiency of the methods do not appear to be tied to this property. Very little variation in iterative performance is also observed relative to the spacing of the abscissa values. Furthermore, none of the methods tested suffered ill effects from having abscissa values outside of the time step. This is seen by comparing Methods 3-6 and 3-7, which have abscissa far outside of the time step, with the reference Methods 3-8 and 3-9, which have the abscissa values wholly within the time step. Both sets of methods were stable, converged, and obtained similar order properties. Those with abscissa values outside of the time step were more efficient, as predicted. This demonstrates that having abscissa values outside of the time step does not preclude a method from being used; however, the reader should be aware that for some problems, this property can cause issues.

5.2.2. Turbulent Flow Around the NACA 0012 Airfoil

It has been observed that the performance of high-order implicit time-marching methods differs when applied to turbulent flows simulated with the Reynolds-averaged Navier-Stokes equations using the Spalart-Allmaras turbulence model. In this case, the governing equations behave more like a system of index one differential algebraic equations, where the stiff source terms in the the turbulent kinetic energy equation are thought to be the source of the algebraic variables [19].

This section considers two-dimensional turbulent flow over a NACA 0012 airfoil at an angle of attack of 30°, Reynolds number of 10^5 and Mach number of 0.2 [19, 20]. Under these conditions, the flow separates near the leading edge of the airfoil creating large vortices in the wake. Similar to the cylinder case, these vortices create smooth regular variations in lift and drag. This problem is simulated using a fourth-order spatial discretization of the

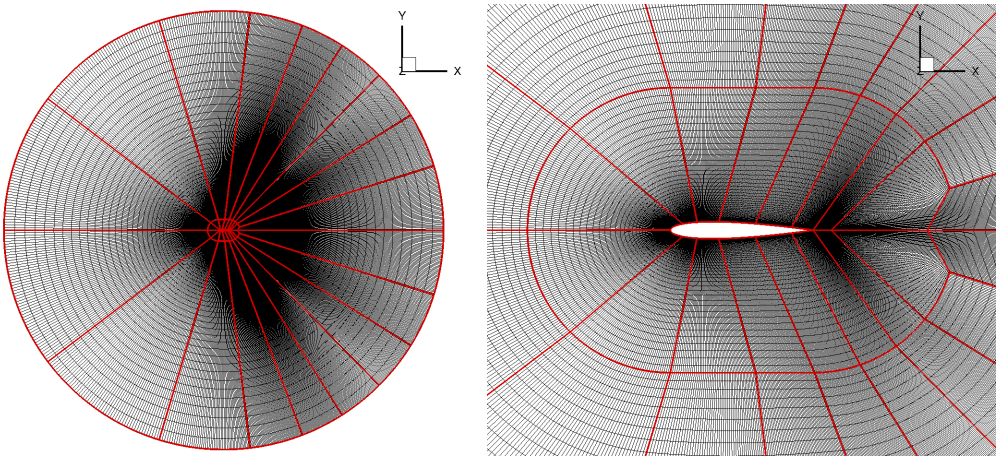

Figure 2: **Turbulent NACA 0012:** Computational grid

Reynolds-Averaged Navier–Stokes equations and a second-order implementation of the turbulence model applied on a grid with $85,000$ cells: 601 nodes on the surface of the airfoil and 101 in the offwall direction. The offwall spacing at the surface is 0.001 times the chord, and 1 in the farfield located a minimum of 20 chord lengths from the surface of the airfoil. Grids are decomposed into 34 blocks and the solution is computed in parallel. Figure 2 shows a plot of the grid.

The maximum aspect ratio is about 15, but in the near-wall region the maximum aspect ratio is similar the the laminar cylinder grid, about 5. The aspect ratio increases toward the farfield as the radial mesh spacing increases faster than the circumferential mesh spacing. This is particularly noticeable in the lines emanating from the upper and lower surface near the trailing edge. In general, the aspect ratio of the grid used for semi-discretization and the stiffness of the resulting ODE will correlate. The aspect ratio does not pose a stability problem, as all methods considered are A-stable and many are L-stable. We would expect that stiffness due to the aspect ratio would influence all of the methods similarly, though L-stable methods or methods for which $M_A(\infty) << 1$ could benefit from the additional damping of high frequency modes.

Similar to the laminar flow over a circular cylinder, the temporal accuracy and efficiency of the methods are evaluated in a temporal convergence study with time steps of sizes from $\Delta t = 2^{-7}$ to $\Delta t = 2^3$. The simulation is run for 40 non-dimensional time units, equal to about 2.58 shedding cycles.

#	Method	$e_{C_L} = 10^{-6}$		$e_{C_D} = 10^{-6}$		Avg.
		p	Lin its	p	Lin its	Lin its/s_{i}
3-2	SDIRK[3,1](3)L_13	1.00	2.27e8	1.00	1.19e8	1.96e2
3-3	SDIRK[3,(1,2,2)](3)L_14	1.00	5.65e7	1.00	3.61e7	1.54e2
3-4	SDIRK[3,1](3)L_SA_al	2.79	7.83e5	2.80	6.54e5	2.40e2
3-5	SDIRK[3,1](3)L_01	1.00	1.15e8	1.00	6.73e7	1.59e2
3-6	SDIRK[3,1](3)AlgL	1.00	1.04e8	1.00	6.71e7	1.13e2
3-7	SDIRK[3,1](3)AlgL_00	1.00	1.83e8	1.00	1.09e8	1.16e2
3-8	SDIRK[3,1](3)Alg_no2	1.00	4.40e7	1.00	3.33e7	1.07e2
3-9	SDIRK[3,1](3)Alg_no	1.00	1.19e8	1.00	7.90e7	1.10e2
3-10	SDIRK[3,(1,2,3,3)](4)L_11	1.14	3.50e7	1.16	2.51e7	3.40e2
3-11	SDIRK[3,1](4)L_SA_5	1.45	4.21e5	1.48	3.73e5	1.95e2
3-12	SDIRK[3,(1,2,2,3)](4)L_SA_7	2.45	4.46e5	2.43	3.87e5	2.11e2
3-13	SDIRK[3,(1,1,2,3)](4)L_SA_ca	2.52	5.51e5	2.56	4.73e5	1.98e2
4-2	SDIRK[4,1](4)L_03	1.00	3.96e8	1.00	2.31e8	1.97e2
4-3	SDIRK[4,(1,2,2,2)](4)L_13	1.00	1.07e7	1.00	9.58e6	1.12e2
4-4	SDIRK[4,1](4)L_01	1.00	9.11e7	1.00	6.31e7	1.43e2
4-5	SDIRK[4,1](4)L_05	1.00	8.93e7	1.00	6.21e7	1.16e2
4-6	SDIRK[4,1](4)L_04	1.00	1.08e8	1.00	7.35e7	1.48e2
4-7	SDIRK[4,1](4)L_02	1.00	4.62e8	1.01	2.66e8	2.29e2
4-8	SDIRK[4,1](4)L_00	1.00	8.00e7	1.00	5.63e7	1.14e2
4-9	SDIRK[4,1](4)L_06	1.00	8.29e7	1.00	5.81e7	1.15e2
5-1	SDIRK[5,1](5)L_02	1.00	5.78e6	1.00	5.25e6	9.23e1
5-2	SDIRK[5,1](5)L_01	1.00	2.13e6	1.00	2.18e6	8.18e1
5-3	SDIRK[5,(1,1,2,2,2)](5)A_co	1.00	5.92e8	1.00	3.25e8	2.52e2
5-4	ESDIRK[5,2](6)A_SA	2.69	5.18e5	3.08	4.42e5	2.27e2
5-5	ESDIRK[5,2](6)L_SA_00	3.22	4.74e5	3.50	4.14e5	2.61e2
5-6	ESDIRK[5,2](6)L_SA_2	3.55	4.60e5	3.81	4.08e5	2.32e2
5-7	ESDIRK[5,2](6)L_SA_01	2.71	4.71e5	2.76	4.11e5	2.32e2
5-8	ESDIRK[5,2](6)L_SA_bm	3.46	4.69e5	3.67	4.19e5	2.62e2
5-9	ESDIRK[5,2](6)L_SA_07	2.97	4.99e5	2.96	4.43e5	2.29e2
5-10	ESDIRK[5,2](6)L_SA_k2c	2.63	5.80e5	2.58	4.83e5	2.27e2

Table 7: **Turbulent flow over the NACA 0012 airfoil:** Stiff and non-stiff convergence rates are presented, along with the number of linear iterations required to obtain an error of 10^{-6}. These values are computed for both lift and drag. The average number of linear iterations required per implicit stage s_{i} is also reported.

Table 7 presents the results of the simulations, including stiff and nonstiff convergence rates computed by a line of best fit through the the smallest three time steps before roundoff error. Individual values are shown for lift and drag, and are interpolated for a prescribed error of 10^{-6}. Also shown are the average number of linear iterations per implicit stage computed from the simulation with the three smallest time step before roundoff error. The methods are organized based on order, followed by the number of stages, then the predicted relative efficiency from Section 4.

In this case, the turbulence model introduces significant stiffness, causing nearly all the methods to suffer from order reduction. The observed order

of convergence for very stiff problems is largely tied to the stage order of the methods. Thus, methods with lower stage order relative to their global order will suffer a greater amount of order reductions. This phenomenon has been observed for unsteady RANS simulations in other articles [19, 20]. The data also shows that non-stiffly-accurate methods only achieve first-order convergence, largely independent of step size (see highlighted rows in the tables). From this we conclude that two of the most important properties for simulation of very stiff problems is stage order and stiff accuracy, something that is well known in the literature [44, 65].

In this case, the relative efficiency of the methods seems to be much more closely tied to the error norm used in the optimization, rather than the LTE coefficient which dominated in the laminar cylinder case. As a result, the efficiency ordering of the stiffly-accurate third and fifth-order methods agree reasonably well with the predictions made in Section 3.

Global and internal algebraic stability play a larger role in the relative efficiency of the methods for this case. This can be seen by considering the fourth-order non-stiffly-accurate SDIRK[4,1] methods. Methods with larger violation in the algebraic stability properties also required a greater number of linear iterations to achieve the prescribed level of accuracy, and vice versa. This is also observed to some extent with the fifth-order ESDIRK[5,2] methods.

The influence of the error norm and algebraic stability properties seem to dominate in this case, though some influence from the internal linear stability remains with the fifth-order Methods 5-5 through 5-7. The spacing of abscissa values plays a limited role in the relative efficiency of the methods, except in the extreme case of Methods 3-6 and 3-7. These methods have very large spacing relative to comparable methods and required noticeably more linear iterations to achieve the desired level of accuracy. As with the laminar cylinder case, we observed no issues with the existence of abscissa values outside the time step. The reader should be aware that this can be an issue for some IVPs and should select a time-marching method appropriate for their application.

Table 8 shows the convergence rates of two solution variables ρ and ρu, as well as the turbulence variable ν. The data shows that solution variables nearly always exhibit higher convergence rates as compared to the turbulence variable. This is characteristic of differential and algebraic variables, respectively. The observed convergence rates are, for the most part, in line with theoretical predictions for singular perturbation problems.

Method	p_ρ	$p_{\rho u}$	p_ν
SDIRK[3,1](3)L_13	2.66	2.60	2.12
SDIRK[3,(1,2,2)](3)L_14	2.71	2.71	2.27
SDIRK[3,1](3)L_SA_sk2	2.74	2.57	2.00
SDIRK[3,1](3)L_01	2.51	2.21	2.35
SDIRK[3,1](3)AlgL	3.08	2.88	1.89
SDIRK[3,1](3)AlgL_00	2.66	2.64	2.47
SDIRK[3,1](3)Alg_no2	2.68	2.65	2.49
SDIRK[3,1](3)Alg_no	2.64	2.44	2.55
SDIRK[3,(1,2,3,3)](4)L_11	2.25	2.23	2.28
SDIRK[3,1](4)L_SA_5	2.73	2.98	2.06
SDIRK[3,(1,2,2,3)](4)L_SA_7	2.54	2.54	2.16
SDIRK[3,(1,1,2,3)](4)L_SA_ca	2.61	2.64	1.96
SDIRK[4,1](4)L_03	2.59	2.56	1.99
SDIRK[4,(1,2,2,2)](4)L_13	2.92	3.15	1.93
SDIRK[4,1](4)L_01	2.58	3.20	2.07
SDIRK[4,1](4)L_05	3.01	3.26	2.3
SDIRK[4,1](4)L_04	2.33	3.19	2.02
SDIRK[4,1](4)L_02	2.59	2.59	2.55
SDIRK[4,1](4)L_07	2.99	3.25	2.17
SDIRK[4,1](4)L_06	3.00	3.28	1.89
SDIRK[5,1](5)L_02	2.55	2.95	1.85
SDIRK[5,1](5)L_01	2.56	2.80	1.73
SDIRK[5,(1,1,2,2,2)](5)A_co	3.17	3.50	2.35
ESDIRK[5,2](6)A_SA	1.99	2.84	2.08
ESDIRK[5,2](6)L_SA_00	2.86	3.10	1.46
ESDIRK[5,2](6)L_SA_2	2.19	3.00	1.68
ESDIRK[5,2](6)L_SA_01	2.26	2.93	1.79
ESDIRK[5,2](6)L_SA_bm	2.20	2.07	1.74
ESDIRK[5,2](6)L_SA_07	2.12	2.48	1.33
ESDIRK[5,2](6)L_SA_k2c	2.52	2.73	1.43

Table 8: **Turbulent flow over the NACA 0012 airfoil:** Stiff convergence rates are presented for flow variables ρ and ρu, as well as the turbuelent variable ν.

6. Summary of Novel Optimized Runge-Kutta Methods

SDIRK[3,(1,2,2)](3)L_14 This method provides a good balance of properties, including a slightly reduced error norm relative to comparable three-stage methods from the literature. SDIRK[3,1](3)L_13 has a lower error norm, but forfeits the increased order on stages two and three, and suffers from a more severe violation in the algebraic stability conditions. SDIRK[3,1](3)L_SA_al is a comparable stiffly-accurate method with slightly higher error norm and violation in the algebraic stability conditions.

SDIRK[3,1](4)L_SA_5 This optimized four-stage method is the most efficient third-order stiffly-accurate scheme considered, and has a good balance of properties. It lowers the relative error norm of the SDIRK-[3,(1,1,2,3)](4)L_SA_ca reference method by more than a factor of two, which is supported by the numerical simulations. Two similar four-stage optimized schemes are SDIRK[3,(1,2,3,3)](4)L_11 and SDIRK-[3,(1,2,2,3)](4)L_SA_7. The first is constructed to have individual stages with higher-order accuracy. This comes at the price of a higher relative error norm, though it is still lower than the reference method listed above. The second method forfeits stiff accuracy in favour of an even smaller relative error norm, nearly 60% smaller than SDIRK[3,1](4)L-_SA_5. This is not fully realized in the numerical simulations, although a noticeable reduction in computational effort is obtained.

Fourth-order methods Many of the most efficient five-stage schemes were rederived using numerical optimization, indicating that they are already minima in the design space. Unfortunately, no four-stage reference methods were identified; however, the optimized methods were used to demonstrate the ability of numerical optimization to achieve a desired balance between competing objectives.

SDIRK[5,1](5)L_02 This optimized five-stage method is more than 40% more efficient then the reference SDIRK[5,(1,1,2,2,2)](5)A_co method based on the relative error norm and is L-stable. It does show a small increase in the violation of the algebraic stability conditions, especially internally, and does not have the increased order for stages 3 through 5. SDIRK[5,1](5)L_01 better balances the violation in the algebraic

stability conditions, and achieves a 34% reduction in error norm relative to the reference.

ESDIRK[5,2](5)L_bm This method provides an improvement in efficient relative to the reference ESDIRK[5,2](6)L_SA_k2c, especially in the turbulent NACA 0012 case. It also improves the spacing in the abscissa and reduces the violation in the algebraic stability constraints. The method does sacrifice a small increase in the violation in the internal linear stability condition, but is globally L-stable.

7. Conclusions

In this article constrained numerical optimization is applied to select the undetermined coefficients in the construction of high-order diagonally-implicit Runge-Kutta methods. The approach first solves the desired order conditions and any stability criteria with a closed form solution. Undetermined coefficients are then set as design variables. Given the multimodal nature of the design space, a Sobol sequence is used to generate a quasi-random set of initial coefficients, which are optimized in parallel. The coefficients are optimized relative to objective functions of accuracy, stability, and computational cost and constrained by the desired stability criteria.

At lower orders the numerical tool rederived many known methods. In this case it is possible to optimize the schemes analytically since there are few undetermined coefficients and the expressions obtained from the order conditions are fairly simple. The results of the numerical optimization at lower orders simply demonstrate the tool's ability to recover known minima. As the order of the schemes is increased, analytical optimization becomes more challenging due to the size and complexity of the expressions resulting from the order conditions. This is where the value of the optimization procedure is realized. The numerical tool is able to discover several novel high-order unconditionally stable schemes which are more efficient than those found in the literature. This is based on the relative L_2-principal error norm of the order conditions one order higher than the methods. The use of numerical optimization is able to ensure that these methods also maintain a balance of stability and solvability. A summary of the novel methods is given in the previous section.

Order properties of the methods derived in this article are verified with numerical simulation of both nonstiff and stiff van der Pol's equation. The

methods are also applied to numerical simulation of vortex shedding in the laminar wake of a circular cylinder and the turbulent wake of a NACA 0012 airfoil at high angle-of-attack. These simulations were conducted to evaluate the numerical performance of the novel methods and to compare them to methods found in the literature. With few exceptions, the simulation results validate the theoretical predictions. In the turbulent case, non-stiffly accurate methods only achieved first-order convergence of lift and drag; however, the solution and turbulence variables exhibit rates more consistent with differential and algebraic variables, respectively.

The results presented in this article demonstrate the value of using numerical optimization in the selection of undetermined coefficients for Runge-Kutta methods. Furthermore, they motivate the application of this approach to more general classes of time-marching methods in the future, such as general linear methods.

Appendix A. List of methods considered

List of reference methods from the literature (* denotes a method rederived using the optimization procedure described in Section 3)

1. *SDIRK[1,1](1)AlgL_SA_IE [23, 37]
2. *SDIRK[2,1](1)Alg_Mid [23]
3. *ESDIRK[2,2](2)Alg_SA_Trap [6, 26, 59]
4. *SDIRK[2,1](2)AlgL_SA_yi [35, 78]
5. *ESDIRK[2,2](3)AlgL_SA_kv [53, 56]
6. *SDIRK[3,1](2)Alg_cr [27]
7. *SDIRK[3,1](3)L_SA_al [1, 70]
8. SDIRK[3,1](3)A_fr [39]
9. SDIRK[3,1](3)Alg_no2 [61]
10. SDIRK[3,1](3)Alg_no [61]
11. EDIRK[3,1](4)L_SA_el2 [35]
12. ESDIRK[3,2](4)L_SA_kc [52, 53]
13. DIRK[3,1](3)AlgL_gsbp [12]
14. SDIRK[3,(1,1,2,3)](4)L_SA_ca [18]
15. *ESDIRK[3,2](4)L_SA_al [2, 70]
16. ESDIRK[3,2](4)L_SA_kv [54, 56]
17. EDIRK[3,2](4)L_SA_el [35]
18. ESDIRK[3,2](4)L_SA_kv [53, 56]
19. ESDIRK[3,2](4)L_SA_hi [47]
20. SDIRK[3,1](4)L_SA_sk [71]
21. ESDIRK[3,2](4)A_SA_wi [77]
22. SDIRK[3,1](4)A_fr2 [39]
23. ESDIRK[3,2](5)L_SA_k2c [53]
24. ESDIRK[3,2](5)L_SA_k2c2I [53]
25. ESDIRK[3,(3,2,3,3,3)](5)L_SA_k2c [53]
26. *SDIRK[4,1](3)Alg_cr [27]
27. DIRK[4,1](3)Alg_co [25]
28. SDIRK[4,1](4)Alg_sp1 [38]
29. SDIRK[4,1](5)AlgL_sp2 [38]
30. SDIRK[4,1](5)L_SA_ha [44, 53]
31. SDIRK[4,(1,2,2,2,4)](5)L_SA_k2c [53]
32. SDIRK[4,1](5)L_SA_ha2 [44, 53]
33. SDIRK[4,1](5)L_SA_sk [70]
34. QSDIRK[4,(1,2,2,2,4)](5)L_SA_k2c [53]
35. ESDIRK[4,2](5)L_SA_kc [52, 53]
36. ESDIRK[4,(2,2,2,3,4)](5)L_SA_kv [53, 56]
37. SDIRK[4,1](6)AlgL_sp3 [38]
38. ESDIRK[4,2](6)L_SA_sk4 [69]
39. ESDIRK[4,2](6)L_SA_k2cA [53]
40. ESDIRK[4,2](6)L_SA_k2cB [53]
41. ESDIRK[4,(2,2,2,2,3,4)](6)L_SA_k2c3I [53]
42. ESDIRK[4,2](6)L_SA_k2c [53]
43. ESDIRK[4,(2,2,3,3,3,4)](6)L_SA_k2c [53]
44. ESDIRK[4,(2,2,3,3,3,4)](6)L_SA_kv2 [56]
45. ESDIRK[4,2](6)L_SA_kc [52, 53]
46. ESDIRK[4,2](6)L_SA_k2cC [53]
47. ESDIRK[4,2](6)L_SA_sk2 [68, 69]
48. ESDIRK[4,2](6)L_SA_k2cD [53]
49. ESDIRK[4,(2,2,2,2,3,4)](6)L_SA_sk [70]
50. QESDIRK[4,(2,2,3,3,3,4)](6)L_SA_k2c [53]
51. ESDIRK[4,2](6)L_SA_k2cE [53]
52. ESDIRK[4,2](6)L_SA_sk6 [71]

53. ESDIRK[4,(2,2,3,2,2,4)](6)L_SA_k2c [53]
54. QESDIRK[4,(2,2,3,3,3,4)](6)L_SA_k2c2 [53]
55. SDIRK[4,1](7)AlgL_sp4 [38]
56. SDIRK[4,1](8)AlgL_sp5 [38]
57. SDIRK[5,(1,1,2,2,2)](5)A_co [25]
58. ESDIRK[5,2](6)L_SA_k2c [53]
59. ESDIRK[5,2](7)L_SA_k2c [53]
60. ESDIRK[5,2](7)L_SA_sk [68]
61. ESDIRK[5,(2,2,3,3,3,4,5)](7)L_SA_kv [56]
62. ESDIRK[5,(2,2,2,2,2,2,4,5)](8)L_SA_k2c [53]
63. ESDIRK[6,2](7)A_k2c [53]

Classes of Runge-Kutta methods optimized, preceded by the number of methods optimized in each class

1. ⟨1⟩ DIRK[4,1](4)L
2. ⟨2⟩ ESDIRK[2,1](3)L_SA
3. ⟨1⟩ ESDIRK[2,2](3)Alg
4. ⟨2⟩ ESDIRK[2,2](3)AlgL_SA
5. ⟨2⟩ ESDIRK[2,2](3)L
6. ⟨1⟩ ESDIRK[2,2](3)L_SA
7. ⟨1⟩ ESDIRK[3,2](3)Alg
8. ⟨2⟩ ESDIRK[3,2](3)A_SA
9. ⟨1⟩ ESDIRK[3,2](4)L
10. ⟨1⟩ ESDIRK[3,2](4)L_SA
11. ⟨2⟩ ESDIRK[4,(2,2,3,3,4)](5)L_SA
12. ⟨1⟩ ESDIRK[4,2](4)Alg
13. ⟨1⟩ ESDIRK[4,2](4)A_SA
14. ⟨1⟩ ESDIRK[4,2](5)L
15. ⟨1⟩ ESDIRK[4,2](5)L(89.55°)_SA
16. ⟨5⟩ ESDIRK[4,2](5)L_SA
17. ⟨2⟩ ESDIRK[5,2](6)L_SA
18. ⟨2⟩ SDIRK[2,(1,2)](2)L
19. ⟨1⟩ SDIRK[2,1](2)Alg
20. ⟨2⟩ SDIRK[2,1](2)AlgL
21. ⟨1⟩ SDIRK[2,1](2)AlgL_SA
22. ⟨1⟩ SDIRK[2,1](2)L_SA
23. ⟨2⟩ SDIRK[3,(1,2,2,3)](4)L_SA
24. ⟨2⟩ SDIRK[3,(1,2,2)](3)L
25. ⟨12⟩ SDIRK[3,(1,2,3,3)](4)L
26. ⟨4⟩ SDIRK[3,(1,2,3,3)](4)L_SA
27. ⟨2⟩ SDIRK[3,(1,2,3)](3)A
28. ⟨2⟩ SDIRK[3,1](3)AlgL
29. ⟨2⟩ SDIRK[3,1](3)L
30. ⟨3⟩ SDIRK[3,1](3)L_SA
31. ⟨4⟩ SDIRK[4,(1,2,2,2)](4)L
32. ⟨1⟩ SDIRK[4,1](3)Alg
33. ⟨8⟩ SDIRK[4,1](4)L
34. ⟨3⟩ SDIRK[4,1](5)L_SA
35. ⟨4⟩ SDIRK[5,1](5)L

Appendix B. Coefficients of Novel Time-Marching Methods

SDIRK[3,(1,2,2)](3)L_14.

$A_{1,1} = 0.435866521508459$
$A_{1,2} = A_{1,3} = 0$
$A_{2,1} = -0.180541824593188$
$A_{2,2} = 0.435866521508459$
$A_{2,3} = 0$
$A_{3,1} = -0.6448674624242866$
$A_{3,2} = 1.049588373659196$
$A_{3,3} = 0.435866521508459$
$b_1 = 0$
$b_2 = 0.5819393784937729$
$b_3 = 0.4180606215062271$
$c_1 = 0.435866521508459$
$c_2 = 0.2553246969152709$
$c_3 = 0.840587432743368$

SDIRK[3,(1,2,3,3)](4)L_11.

$A_{1,1} = 0.2236468442071308$
$A_{1,2} = A_{1,3} = A_{1,4} = 0$
$A_{2,1} = -0.09263755605253625$
$A_{2,2} = 0.2236468442071308$
$A_{2,3} = A_{2,4} = 0$
$A_{3,1} = 0.029090502594485$
$A_{3,2} = -0.1674714344479084$
$A_{3,3} = 0.2236468442071308$
$A_{3,4} = 0$
$A_{4,1} = 0.2793910597960622$
$A_{4,2} = 1.172529025624291$
$A_{4,3} = -0.8748372875708956$
$A_{4,4} = 0.2236468442071308$
$b_1 = 0$
$b_2 = 1.351040830480596$
$b_3 = -0.8443333686807888$
$b_4 = 0.4932925382001925$
$c_1 = 0.2236468442071308$
$c_2 = 0.1310092881545946$
$c_3 = 0.0852659123537074$
$c_4 = 0.8007296420565881$

SDIRK[3,1](4)L_SA_5.

$A_{1,1} = 0.2236509951645569$
$A_{1,2} = A_{1,3} = A_{1,4} = 0$
$A_{2,1} = 0.3210161240223837$
$A_{2,2} = 0.2236509951645569$
$A_{2,3} = A_{2,4} = 0$
$A_{3,1} = -0.9231923320092694$
$A_{3,2} = 1.475417379665253$
$A_{3,3} = 0.2236509951645569$
$A_{3,4} = 0$
$A_{4,1} = 0.4108468452988502$
$A_{4,2} = 0.4287104001078981$
$A_{4,3} = -0.06320824057130515$
$A_{4,4} = 0.2236509951645569$
$b_1 = 0.4108468452988502$
$b_2 = 0.4287104001078981$
$b_3 = -0.06320824057130515$
$b_4 = 0.2236509951645569$
$c_1 = 0.2236509951645569$
$c_2 = 0.5446671191869406$
$c_3 = 0.7758760428205402$
$c_4 = 1$

SDIRK[3,(1,2,2,3)](4)L_SA_7.

$A_{1,1} = 0.2236468426706971$
$A_{1,2} = A_{1,3} = A_{1,4} = 0$
$A_{2,1} = -0.09263755541612455$
$A_{2,2} = 0.2236468426706971$
$A_{2,3} =_{2,4}= 0$
$A_{3,1} = -0.3390239162242422$
$A_{3,2} = 0.5361289668097047$
$A_{3,3} = 0.2236468426706971$
$A_{3,4} = 0$
$A_{4,1} = 0$
$A_{4,2} = 0.1735985747713019$
$A_{4,3} = 0.602754582558001$
$A_{4,4} = 0.2236468426706971$
$b_1 = 0$
$b_2 = 0.1735985747713019$
$b_3 = 0.602754582558001$
$b_4 = 0.2236468426706971$
$c_1 = 0.2236468426706971$
$c_2 = 0.1310092872545725$
$c_3 = 0.4207518932561596$
$c_4 = 1$

SDIRK[4,(1,2,2,2)](4)L_13.

$A_{1,1} = 0.5728160624821349$
$A_{1,2} = A_{1,3} = A_{1,4} = 0$
$A_{2,1} = -0.2372681818252545$
$A_{2,2} = 0.5728160624821349$
$A_{2,3} = A_{2,4} = 0$
$A_{3,1} = -0.843659473560103$
$A_{3,2} = 0.9783006024430643$
$A_{3,3} = 0.5728160624821349$
$A_{3,4} = 0$
$A_{4,1} = -0.6504189474582887$
$A_{4,2} = 0.3710566153293516$
$A_{4,3} = 0.1337302071646674$
$A_{4,4} = 0.5728160624821349$
$b_1 = 0$
$b_2 = 2.001951626974973$
$b_3 = 0.914347024151788$
$b_4 = -1.916298651126761$
$c_1 = 0.5728160624821349$
$c_2 = 0.3355478806568805$
$c_3 = 0.7074571913650962$
$c_4 = 0.4271839375178652$

SDIRK[4,1](4)L_05.

$A_{1,1} = 0.5728160624821349$
$A_{1,2} = A_{1,3} = A_{1,4} = 0$
$A_{2,1} = -0.4506409404207292$
$A_{2,2} = 0.5728160624821349$
$A_{2,3} = A_{2,4} = 0$
$A_{3,1} = -0.417982144232982$
$A_{3,2} = 0.6303293042757349$
$A_{3,3} = 0.5728160624821349$
$A_{3,4} = 0$
$A_{4,1} = 0.6974938714633269$
$A_{4,2} = -0.4925759495246813$
$A_{4,3} = -0.3505500469029152$
$A_{4,4} = 0.5728160624821349$
$b_1 = -0.426559400640419$
$b_2 = 0.2441815104885498$
$b_3 = 0.5849900719458051$
$b_4 = 0.5973878182060641$
$c_1 = 0.5728160624821349$
$c_2 = 0.1221751220614057$
$c_3 = 0.7851632225248877$
$c_4 = 0.4271839375178653$

SDIRK[5,1](5)L_02.

$A_{1,1} = 0.2780538411364523$
$A_{1,2} = A_{1,3} = A_{1,4} = A_{1,5} = 0$
$A_{2,1} = 0.5884293738285219$
$A_{2,2} = 0.2780538411364523$
$A_{2,3} = A_{2,4} = A_{2,5} = 0$
$A_{3,1} = 0.4757737281134862$
$A_{3,2} = -0.1649111223966794$
$A_{3,3} = 0.2780538411364523$
$A_{3,4} = A_{3,5} = 0$
$A_{4,1} = -0.1430556691639315$
$A_{4,2} = 0.2168859326308357$
$A_{4,3} = -0.3518841046033565$
$A_{4,4} = 0.2780538411364523$
$A_{4,5} = 0$
$A_{5,1} = 1.580366530916478$
$A_{5,2} = 0.1469597740924957$
$A_{5,3} = -0.6778647342704042$
$A_{5,4} = -0.605569255223904$
$A_{5,5} = 0.2780538411364523$
$b_1 = 0.3632241891213434$
$b_2 = 0.3363544171822351$
$b_3 = 0.3182542934848578$
$b_4 = 0.09279380620749932$
$b_5 = -0.1106267059959357$
$c_1 = 0.2780538411364523$
$c_2 = 0.8664832149649742$
$c_3 = 0.5889164468532591$
$c_4 = 0$
$c_5 = 0.7219461566511176$

ESDIRK[5,2](6)A_SA.

$A_{1,1} = A_{1,2} = A_{1,3} = 0$
$A_{1,4} = A_{1,5} = A_{1,6} = 0$
$A_{2,1} = 0.246505193307038$
$A_{2,2} = 0.246505193307038$
$A_{2,3} = A_{2,4} = A_{2,5} = A_{2,6} = 0$
$A_{3,1} = 0.2450410672405718$
$A_{3,2} = 0.4973855759477314$
$A_{3,3} = 0.246505193307038$
$A_{3,4} = A_{3,5} = A_{3,6} = 0$
$A_{4,1} = 0.2564937950047032$
$A_{4,2} = 0.03908988375200104$
$A_{4,3} = -0.008556539796649578$
$A_{4,4} = 0.246505193307038$
$A_{4,5} = A_{4,6} = 0$
$A_{5,1} = 0.04501659096048612$
$A_{5,2} = 1.067115793643888$
$A_{5,3} = 0.06024953770808324$
$A_{5,4} = -1.154386154396951$
$A_{5,5} = 0.246505193307038$
$A_{5,6} = 0$
$A_{6,1} = 0.04357627047518315$
$A_{6,2} = -3.24634446857275$
$A_{6,3} = -0.1374502416258243$
$A_{6,4} = 3.37177845072737$
$A_{6,5} = 0.7219347956889822$
$A_{6,6} = 0.246505193307038$
$b_1 = 0.04357627047518315$
$b_2 = -3.24634446857275$
$b_3 = -0.1374502416258243$
$b_4 = 3.37177845072737$
$b_5 = 0.7219347956889822$
$b_6 = 0.246505193307038$
$c_1 = 0$
$c_2 = 0.493010386614076$
$c_3 = 0.9889318364953411$
$c_4 = 0.5335323322670926$
$c_5 = 0.2645009612225441$
$c_6 = 1$

ESDIRK[5,2](6)L_SA_bm.

$A_{1,1} = A_{1,2} = A_{1,3} = 0$
$A_{1,4} = A_{1,5} = A_{1,6} = 0$
$A_{2,1} = 0.2780538411364523$
$A_{2,2} = 0.2780538411364523$
$A_{2,3} = A_{2,4} = A_{2,5} = A_{2,6} = 0$
$A_{3,1} = 0.3262449081464093$
$A_{3,2} = 0.3838598306191588$
$A_{3,3} = 0.2780538411364523$
$A_{3,4} = A_{3,5} = A_{3,6} = 0$
$A_{4,1} = 0.2991093048633836$
$A_{4,2} = 0.1491319274791011$
$A_{4,3} = -0.03781208693479417$
$A_{4,4} = 0.2780538411364523$
$A_{4,5} = A_{4,6} = 0$
$A_{5,1} = -0.2994138907017297$
$A_{5,2} = -0.7626765721782363$
$A_{5,3} = -0.1690593157465009$
$A_{5,4} = 0.8880327042046392$
$A_{5,5} = 0.2780538411364523$
$A_{5,6} = 0$
$A_{6,1} = 0.5920973606196398$
$A_{6,2} = 0.4332458780105385$
$A_{6,3} = -0.1688012973030839$
$A_{6,4} = 0.1845407853751282$
$A_{6,5} = -0.319136567838675$
$A_{6,6} = 0.2780538411364523$
$b_1 = 0.5920973606196398$
$b_2 = 0.4332458780105385$
$b_3 = -0.1688012973030839$
$b_4 = 0.1845407853751282$
$b_5 = -0.319136567838675$
$b_6 = 0.2780538411364523$
$c_1 = 0$
$c_2 = 0.5561076822729046$
$c_3 = 0.9881585799020205$
$c_4 = 0.6884829865441429$
$c_5 = -0.06506323328537517$
$c_6 = 1$

References

[1] R. Alexander, Diagonally Implicit Runge-Kutta Methods for Stiff O.D.E.'s, SIAM Journal on Numerical Analysis 14 (1977) 1006–1021.

[2] R. Alexander, Design and Implementation of DIRK Integrators for Stiff Systems, Applied Numerical Mathematics 46 (2003) 1–17.

[3] R.K. Alexander, J.J. Coyle, RungeKutta Methods and Differential-Algebraic Systems, SIAM Journal on Numerical Analysis 27 (1990) 736752.

[4] S.R. Allmaras, F.T. Johnson, P.R. Spalart, Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model, 7^{th} International Conference on Computational Fluid Dynamics ICCFD7-1902 (2012).

[5] R. Alt, Méthodes A-stables pour l'intégration des systèmes différentiels mal conditionnés, Ph.D. thesis, Université Paris, 1971.

[6] F. Bashforth, J.C. Adams, An Attempt to Test the Theories of Capillary Action: By Comparing the Theoretical and Measured Forms of Drops of Fluid. With an Explanation of the Method of Integration Employed in Constucting the Tables which Give the Theoretical Forms of Such Drops, Cambridge University Press (1883).

[7] J. Berland, C. Bogey, C. Bailly, Low-dissipation and low-dispersion fourth-order Runge–Kutta algorithm, Computers & Fluids 35 (2006) 1459–1463.

[8] T.A. Bickart, W.B. Rubin, Composite Multistep Methods and Stiff Stability, Composite Multistep Methods and Stiff Stability, Springer US, Boston, MA, 1974, pp. 21–36.

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

[10] P.D. Boom, High-order Implicit Time-Marching Methods for Unsteady Fluid Flow Simulation, Ph.D. thesis, University of Toronto Institute for Aerospace Studies, 4925 Dufferin Street, Toronto, Ontario, Canada M3H 5T6, 2015.

[11] P.D. Boom, D.W. Zingg, High-Order Implicit Time Integration for Unsteady Compressible Fluid Flow Simulations, 21st AIAA Computational Fluid Dynamics Conference AIAA-2013-2831 (2013).

[12] P.D. Boom, D.W. Zingg, High-Order Implicit Time-Marching Methods Based on Generalized Summation-By-Parts Operators, SIAM Journal on Scientific Computing 37 (2015) A2682–A2709.

[13] K. Burrage, J.C. Butcher, Stability Criteria for Implicit Runge-Kutta Methods, SIAM Journal on Numerical Analysis 16 (1979) pp. 46–57.

[14] K. Burrage, J.C. Butcher, Non-linear Stability of a General Class of Differential Equation Methods, BIT Numerische Mathematik 20 (1980) 185–203.

[15] J.C. Butcher, Coefficients for the Study of Runge-Kutta Integration Processes, Journal of the Austrailian Mathematical Society 3 (1963) 185–201.

[16] J.C. Butcher, Implicit Runge-Kutta Processes, Mathematics of Computation 18 (1964) 50–64.

[17] J.C. Butcher, A Stability Property of Implicit Runge-Kutta Methods, BIT Numerische Mathematik 15 (1975) 358–361.

[18] F. Cameron, M. Palmroth, R. Pich, Quasi stage order conditions for SDIRK methods, Applied Numerical Mathematics 42 (2002) 61–75.

[19] M.H. Carpenter, C.A. Kennedy, H. Bijl, S.A. Viken, V.N. Vatsa, Fourth-Order Runge-Kutta Schemes for Fluid Mechanics Applications, Journal of Scientific Computing 25 (2005) 157–194.

[20] M.H. Carpenter, S.A. Viken, E. Nielsen, The Temporal Efficiency of Higher Order Schemes, 41st AIAA Aerospace Sciences Meeting and Exhibit AIAA-2003-0086 (2003).

[21] J.R. Cash, On the Integration of Stiff Systems of O.D.E.s Using Extended Backward Differentiation Formulae, Numerische Mathematik 34 (1980) 235–246.

[22] J.R. Cash, The Integration of Stiff Initial Value Problems in ODEs Using Modified Extended Backward Differentiation Formulae, Computers & Mathematics with Applications 9 (1983) 645–657.

[23] A.L. Cauchy, C. Gilain, Résumé des Leçons Données a l'École Royale Polytechniue. Suite du Calcul Infinitésimal (1824), in: Équations Différentielles Ordinaires: Cours Inédit: Fragment, Études vivantes, Paris, 1981.

[24] O. Chernukhin, D.W. Zingg, Multimodality and Global Optimization in Aerodynamic Design, AIAA Journal 51 (2013) 1342–1354.

[25] G.J. Cooper, A. Sayfy, Semi-explicit A-stable Runge-Kutta Methods, Mathematics of Computation 33 (1979) 541–556.

[26] J. Crank, P. Nicolson, A Practical Method for Numerical Evaluation of Solutions of Partial Differential Equations of the Heat Conduction Type, Mathematical Proceedings of the Cambridge Philosophical Society 43 (1947) 50–67.

[27] M. Crouzeix, Sur L'Approximation des Équations Différentielle Opérationelles Linéaires par des Méthodes de Runge-Kutta, Ph.D. thesis, Université de Paris, 1975.

[28] M. Crouzeix, Sur la B-stabilité des Méthodes de Runge-Kutta, Numerische Mathematik 32 (1979) 75–82.

[29] C.F. Curtiss, J.O. Hirschfelder, Integration of Stiff Equations, Proceedings of the National Academy of Sciences of U.S. 38 (1952) 235–243.

[30] G.G. Dahlquist, A Special Stability Problem for Linear Multistep Methods, BIT Numerische Mathematik 3 (1963) 27–43.

[31] K. Dekker, J. Verwer, Stability of Runge-Kutta Methods for Stiff Nonlinear Differential Equations, CWI monograph, 1984.

[32] J. Donelson III, E. Hansen, Cyclic Composite Multistep Predictor-Corrector Methods, SIAM Journal on Numerical Analysis 8 (1971) 137–21.

[33] B.L. Ehle, High Order A-Stable Methods for the Numerical Solution of Systems of DEs, BIT Numerische Mathematik 8 (1968) 276–278.

[34] B.L. Ehle, On Padé Approximations to the Exponential Function and A-stable Methods for the Numerical Solution of Initial Value Problems, Technical Report CS-RR-2010, Department of Applied Analysis and Computer Science, University of Waterloo, 1969.

[35] P. Eliasson, P. Weinerfelt, High-order implicit time integration for unsteady turbulent flow simulations, Computers & Fluids 112 (2015) 35–49.

[36] R. England, Some hybrid implicit stiffly stable methods for ordinary differential equations, Some hybrid implicit stiffly stable methods for ordinary differential equations, Springer Berlin Heidelberg, Berlin, Heidelberg, 1982, pp. 147–158.

[37] L. Euler, Institutionum Calculi Integralis. Volumen Primum, Petropoli, Impensis Academiae Imperialis Scientiarum, 1768. Reprint: Opera Omnia: Vol. XI.

[38] L. Ferracina, M. Spijker, Strong Stability of Singly-Diagonally-Implicit Runge–Kutta Methods, Applied Numerical Mathematics 58 (2008) 1675–1686.

[39] J. Franco, I. Gómez, L. Rández, SDIRK Methods for Stiff ODEs with Oscillating Solutions, Journal of Computational and Applied Mathematics 81 (1997) 197–209.

[40] P.E. Gill, W. Murray, M.H. Wright, Practical optimization, Academic Press, 1981.

[41] S. Gill, A process for the step-by-step integration of differential equations in an automatic digital computing machine, Mathematical Proceedings of the Cambridge Philosophical Society 47 (1951) 96–108.

[42] A. Guillou, J.L. Soulé, La Résolution Numérique des Problèmes Différentiels aux Conditions Initiales par des Méthodes de Collocation, R.I.R.O. 3 (1969) 17–44.

[43] E. Hairer, C. Lubich, M. Roche, Error of Runge-Kutta Methods for Stiff Problems Studied Via Differential Algebraic equations, BIT Numerische Mathematik 28 (1988) 678–700.

[44] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II, Springer, Berlin, 1991.

[45] E. Hansen, Cyclic composite multistep predictor-corrector methods, in: Proceedings of the 1969 24th National Conference, ACM '69, ACM, New York, NY, USA, 1969, pp. 135–139.

[46] J.E. Hicken, D.W. Zingg, A parallel Newton-Krylov solver for the Euler equations discretized using simultaneous approximation terms, AIAA Journal 46 (2008) 2773–2786.

[47] I. Higueras, T. Roldn, Starting algorithms for a class of RK methods for index-2 DAEs, Computers & Mathematics with Applications 49 (2005) 1081–1099.

[48] M. Hosea, L. Shampine, Analysis and implementation of TR-BDF2, Applied Numerical Mathematics 20 (1996) 2137.

[49] F.Q. Hu, M.Y. Hussaini, J.L. Manthey, Low-dissipation and Low-dispersion Runge-Kutta Schemes for Computational Acoustics, Journal of Computational Physics 124 (1996) 177–191.

[50] Z. Jackiewicz, General Linear Methods for Ordinary Differential Equations, Wiley, 2009.

[51] S. Joe, S.J. Wright, Remarks on Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator, ACM Transactions on Mathematical Software 29 (2003) 49–57.

[52] C.A. Kennedy, M.H. Carpenter, Additive Runge-Kutta Schemes for Convection-Diffusion-Reaction Equations, Applied Numerical Mathematics 44 (2003) 139–181.

[53] C.A. Kennedy, M.H. Carpenter, Diagonally Implicit Runge-Kutta Methods for Ordinary Differential Equations. A Review, Technical Report NASA/TM 2016-219173, 2016.

[54] M.R. Kristensen, J.B. Jørgensen, P.G. Thomsen, S.B. Jørgensen, An ESDIRK Method with Sensitivity Analysis Capabilities, Computers & Chemical Engineering 28 (2004) 2695–2707.

[55] M.A. Kurdi, Stable high order methods for time discretization of stiff differential equations, Ph.D. thesis, University of California, Berkeley, 1974.

[56] A. Kværnø, Singly Diagonally Implicit Runge–Kutta Methods with an Explicit First Stage, BIT Numerische Mathematik 44 (2004) 489–502.

[57] H. Lomax, T.H. Pulliam, D.W. Zingg, Fundamentals of Computational Fluid Dynamics, Scientific Computation, Springer, 2001.

[58] R.H. Merson, An operational method for the study of integration processes, in: Proceedings Symposium on Data Processing, Weapons Research Establishment, Salisbury, Austrailia, 1957, pp. 110–125.

[59] F.R. Moulton, New Methods in Exterior Ballistics, Cambridge University Press (1926).

[60] S.P. Nørsett, Semi-Explicit Runge-Kutta Methods, Technical Report 6/74, University of Trondheim, 1974.

[61] S.P. Nørsett, P.G. Thomsen, Embedded SDIRK-Methods of Basic Order Three, BIT Numerische Mathematik 24 (1984) 634–646.

[62] M. Osusky, D.W. Zingg, Parallel Newton-Krylov-Schur Solver for the Navier-Stokes Equations Discretized Using Summation-By-Parts Operators, AIAA Journal 51 (2013) 2833–2851.

[63] M. Parsani, D.I. Ketcheson, W. Deconinck, Optimized Explicit Runge-Kutta Schemes for the Spectral Difference Method Applied to Wave Propagation Problems, SIAM Journal on Scientific Computing 35 (2013) A957–A986.

[64] P. Prince, J. Dormand, High Order Embedded Runge-Kutta Formulae, Journal of Computational and Applied Mathematics 7 (1981) 67–75.

[65] A. Prothero, A. Robinson, On the Stability and Accuracy of One-Step Methods for Solving Stiff Systems of Ordinary Differential Equations, Mathematics of Computation 28 (1974) 145–162.

[66] G.Y. Psihoyios, Advanced Step-point Methods for the Solution of Initial Value Problem, Ph.D. thesis, University of London - Imperial College of Science and Technology, 1995.

[67] G.Y. Psihoyios, J.R. Cash, A Stability Result for General Linear Methods with Characteristic Function Having Real Poles Only, BIT Numerische Mathematik 38 (1998) 612–617.

[68] L.M. Skvortsov, Diagonal Implicit FSAL Runge-Kutta Methods for Stiff and Differential-Algebraic Systems, Journal of Mathematical Modeling 14 (2002) 3–17.

[69] L.M. Skvortsov, Accuracy of Runge–Kutta Methods Applied to Stiff Problems, Computational Mathematics and Mathematical Physics 43 (2003) 1320–1330.

[70] L.M. Skvortsov, Diagonally Implicit Runge-Kutta Methods for Stiff Problems, Computational Mathematics and Mathematical Physics 46 (2006) 2110–2123.

[71] L.M. Skvortsov, Diagonally Implicit Runge–Kutta Methods for Differential Algebraic Equations of Indices Two and Three, Computational Mathematics and Mathematical Physics 50 (2010) 993–1005.

[72] H.M. Sloate, T.A. Bickart, A-Stable Composite Multistep Methods., Journal of the ACM 20 (1973) 7–26.

[73] I.M. Sobol, Distribution of Points in a cube and Approximate Evaluation of Integrals, (in Russian) Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki 7 (1967) 784–802. (in English) U.S.S.R. Computational Mathematics and Mathematical Physics, 7 (1967), pp. 86-112.

[74] W. Squire, G. Trapp, Using complex variables to estimate derivatives of real functions, SIAM Review 40 (1998) 110–112.

[75] M. Tabesh, D.W. Zingg, Efficient Implicit Time-Marching Methods Using a Newton-Krylov Algorithm, 47th AIAA Aerospace Sciences Meeting AIAA-2009-164 (2009).

[76] R. Williams, K. Burrage, I. Cameron, A new index 2 Runge-Kutta method for the simulation of batch and discontinuous processes , Computers and Chemical Engineering 2–7 (2000) 625–630.

[77] R. Williams, K. Burrage, I. Cameron, M. Kerr, A Four-Stage Index 2 Diagonally Implicit Runge–Kutta Method, Applied Numerical Mathematics 40 (2002) 415–432.

[78] W. Ying, D. Rose, C. Henriquez, Efficient fully implicit time integration methods for modeling cardiac dynamics, IEEE Transactions on Biomedical Engineering 55 (2008) 2701–2711.

[79] D.W. Zingg, T.T. Chisholm, Runge–Kutta methods for linear ordinary differential equations, Applied Numerical Mathematics 31 (1999) 227–238.

[80] D.W. Zingg, H. Lomax, H. Jurgens, High-accuracy finite-difference schemes for linear wave propagation, SIAM Journal on Scientific Computing 17 (1996) 328–346.