Tenth International Conference on Computational Fluid Dynamics (ICCFD10), Barcelona, Spain, July 9-13, 2018 **ICCFD10-2018-111**

Numerical optimization of fourth-order diagonally-implicit multistep Runge-Kutta methods

P.D. Boom* and D.W. Zingg*
Corresponding author: pieter.boom@mail.utoronto.ca
* University of Toronto Institute for Aerospace Science, Canada.

Abstract: This article presents constrained numerical optimization of fourth-order L-stable multistep Runge-Kutta (MRK) methods. The methods are optimized relative to composite objective functions accounting for accuracy, internal stability, conditioning, and computational cost. Global stability properties and bounds on the coefficients are enforced through linear and nonlinear constraints. The relative benefits of increasing the number of stages versus the number of steps is discussed, along with comparisons to implicit linear multistep (LM) and implicit Runge-Kutta (RK) methods. With the chosen objective function, the optimized MRK methods are not expected to be the most efficient. However, they do obtain a combination of properties that neither the LM or RK methods can. Furthermore, when applied to laminar flow over a circular cylinder, the optimized L-stable fourth-order two-step four-stage stiffly-accurate singly-diagonally-implicit multistep Runge-Kutta method SDIMRK[4,2](4,2)L_SA_0 was the most efficient. This is partly due to the accuracy being more related to the local truncation error for this case, rather than the L_2-principal error norm for which the methods were optimized. Simulation of van der Pol's equation also confirms the order properties of all methods for nonstiff and stiff problems.

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

1 Introduction

Stiff initial-value problems (IVPs) arise frequently in computational fluid dynamics (CFD), either from the underlying physics governing the flow or through numerical approximations. These IVPs are characterized by high-frequency parasitic modes, in addition to the relevant driving modes [1]. Parasitic modes are small, but must remain stable. The numerical methods applied to solve these IVPs must therefore balance accuracy of the driving modes, the stability of the parasitic modes, and computational cost to be useful and efficient.

Traditionally, stiff IVPs arising in CFD are solved using either low-order implicit linear multistep (LM) or higher-order implicit Runge-Kutta (RK) methods. Unconditionally stable LM methods are known to be robust and efficient over a single time step. New solution points are created using solution and function values from previous time steps, as well as a single new implicitly defined function value. However, unconditionally stable LM methods are limited to second-order [2]. As the desired accuracy becomes increasingly stringent, the number of time steps required to obtain a result can become prohibitive.

Fortunately, unconditionally stable implicit RK methods can be constructed for arbitrary orders of accuracy [3]. This can greatly reduce the number of time steps required to achieve a prescribed level of accuracy. Implicit RK methods construct a new solution point from the solution value at the previous time step, as well as multiple new implicitly defined function values computed at off-step locations. The off-step solution points are called stages. Therefore, over one time step RK methods are more computationally expensive than LM methods. When the desired level of accuracy is stringent, the reduction in total number of time steps can outweigh the additional cost per time step.

Another, less common, alternative is to use generalizations of LM and RK methods which combine the properties of both classes. Many of these methods can be described by the umbrella class of Multistep

Runge-Kutta (MRK) methods [4]. These methods make use of both multiple steps and multiple stages. The interest in this article is to be able to achieve high-order accuracy and unconditional stability, while reducing computational cost per time step. Another advantage of implicit MRK methods is the ability to achieve higher-stage order than implicit RK methods. This is a particular benefit for IVPs which exhibit order reduction, such as differential algebraic problems. In this paper, we will focus on the construction and evaluation of MRK methods which use past solution values only. Methods that make use of past function values, will be left to a future paper.

The construction of implicit time-marching methods often begins with solving the desired order conditions. Generally, this will only determine a subset of the coefficients. As the size, order and complexity of the methods grow, it becomes increasingly difficult to assign values to these undetermined coefficients which maximize computational efficiency and satisfy the desired stability criteria. In this article we apply constrained numerical optimization to the construction of new fourth-order L-stable stiffly-accurate singly-diagonally-implicit MRK methods. The undetermined coefficients are optimized relative to objective functions which account for accuracy, internal stability, conditioning, and computational cost. The desired global stability properties are enforced with nonlinear constraints. This is similar to the work done for implicit RK methods [5, 6, 7], extending it to the class of implicit MRK methods. It also broadens the work presented by Boom [6]. The properties of the novel methods are verified through numerical solution of van der Pol's equation and laminar flow over a circular cylinder.

Section 2 gives a short review of implicit MRK methods and their properties. These properties are further discussed in the description of the optimization procedure presented in Section 3. The novel optimized implicit MRK methods are presented in Section 4 and compared numerically in Section 5. Conclusions are presented in Section 6.

2 Multistep Runge-Kutta Methods

A multistep Runge-Kutta (MRK) method applied to the initial-value problem (IVP)

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

approximates the next solution point as

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

where h is the time step size, n is the time step index such that $y^{[n]} \approx \mathcal{Y}(t^{[n]})$ for $t^{[n]} = t_0 + hn$, r is the number of past solution values used (number of steps), and s is the number of internal stage approximations

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

The coefficients of a particular scheme are defined by the matrices A and U, and the vectors $\mathbf{b}$ and $\mathbf{v}$. The internal abscissa vector $\mathbf{c}$ is determined from these coefficients using the first stage order condition described in Section 2.1. Note that this definition can easily be extended to systems of differential equations using Kronecker products.

In this article we are interested in singly-diagonally-implicit MRK methods. This places additional restrictions of the A coefficient matrix: that it be lower triangular and have a constant diagonal coefficient. A lower triangular A coefficient matrix makes the solution to each internal stage approximation independent of subsequent stages. Therefore, the stages can be solved sequentially. The restriction on the diagonal coefficient steps can improve the efficiency of Newton-type processes commonly used to solve the resulting nonlinear system at each stage. In this case, the residual equation for an intermediate stage (3) can be written as

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

and the Jacobian with respect to the given internal stage approximation Y_k is

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

where I is the identity matrix. If the diagonal entries of the A coefficient matrix are equal, then when 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, the Jacobian can be approximated as constant over one or more time steps. Thus operations on the Jacobian, such as an LU decomposition or the construction of a preconditioner, can be reused in each stage without significant reduction in convergence [8, 9].

Another restriction that can be placed on the coefficients of a given MRK method is known as stiff accuracy. The algebraic conditions for this restriction are

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

implying that $y^{[n+1]} = Y_s$. Stiff accuracy comes from a well-known study of order reduction by Prothero and Robinson [10]. Order reduction is the phenomenon in the solution of stiff IVPs where high-order implicit time-marching methods exhibit lower order convergence than predicted by theory. Stiff accuracy does not eliminate order-reduction but does influence both the stability and convergence rate observed. This property also plays a role in the convergence rates of differential algebraic and singular perturbation problems, for example van der Pol's equation (See *e.g.* [11] and Section 5).

2.1 Order conditions

A time-marching method is said to be of order p if the local error is:

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

Using Butcher series and the following augmented matrices

$$B = \begin{bmatrix} \mathbf{b}^T \\ \mathbf{0}_{(r-1)\times s} \end{bmatrix} \quad \text{and} \quad V = \begin{bmatrix} \multicolumn{2}{c}{\mathbf{b}^T} \\ I_{(r-1)\times(r-1)} & \mathbf{0}_{(r-1)\times 1} \end{bmatrix}. \tag{8}$$

it can be shown that an MRK method will be of order p if [12]:

$$\mathtt{O}(\mathtt{t}) = (V - I)\mathtt{Q}(\mathtt{t}) - \sum_{\mathtt{u}\subset\mathtt{t}} \binom{\rho(\mathtt{t})}{\rho(\mathtt{u})} \frac{\alpha(\mathtt{u},\mathtt{t})}{\alpha(\mathtt{t})} \mathtt{Q}(\mathtt{u}) + \rho(\mathtt{t}) B \prod_{k=1}^{m} \mathtt{Y}(\mathtt{t}_k) = 0, \quad \forall \mathtt{t} \text{ such that } 0 \le \rho(\mathtt{t}) \le p, \tag{9}$$

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. 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 linear map $\mathtt{Q}(\mathtt{t}) : \mathtt{t} \to \mathbb{R}$ is equal to $[0^{\rho(\mathtt{t})}, 1^{\rho(\mathtt{t})}, \ldots, (r-1)^{\rho(\mathtt{t})}]$ for MRK methods, $\alpha(\mathtt{t})$ is the number of monotonic labellings of the tree $\mathtt{t}$ starting with the root, $\alpha(\mathtt{t},\mathtt{u})$ is the number of monotonic labellings such that the subtree $\mathtt{u}$, with the same root as $\mathtt{t}$, is labelled first. Here $\binom{\blacksquare}{\blacksquare}$ represents the binomial coefficient and the product $\prod$ is computed element wise. Finally, the contribution from the stages is

$$\mathtt{Y}(\mathtt{t}) = U\mathtt{Q}(\mathtt{t}) + \rho(\mathtt{t}) A \prod_{k=1}^{m} \mathtt{Y}(\mathtt{t}_k) \tag{10}$$

This expression can also be used to define the stage order q of an MRK method:

$$\mathbf{c}^{\rho(\mathtt{t})} = U\mathtt{Q}(\mathtt{t}) + \rho(\mathtt{t}) A \mathbf{c}^{\rho(\mathtt{t})-1}, \quad \forall \mathtt{t} \text{ such that } \rho(\mathtt{t}) \le q. \tag{11}$$

These order and stage order conditions are not necessarily easy to solve, but they form a necessary and sufficient set of conditions for order p and stage order q, respectively.

Simplifying order conditions: In this article we apply the full order conditions in the derivation and optimization of new methods. However, the extent to which the methods satisfy the simplifying conditions of Burrage [13] is presented for reference. These conditions were designed to ease the construction of new higher-order methods and form a sufficient but not necessary set of requirements for order p. Here we use the additional freedom afforded by the full order conditions in the optimization to improve efficiency and robustness. In future, the simplifying conditions could be used to initialize coefficients or as part of an objective function since they are already implemented in the optimization package described in Section 3. The simplifying order conditions are summarized in the following theorem from Burrage [14] and simplified for MRK methods in Hairer and Wanner [11].

Theorem 1. *If the coefficients A, U, $\mathbf{b}$, $\mathbf{v}$, and $\mathbf{c}$ of a MRK method satisfy the following conditions:*

$$\mathbf{B}(p): \quad j\mathbf{b}^T\mathbf{c}^{j-1} + \mathbf{v}^T\mathtt{Q}(\tau^p) = 1, \quad j \in [1,p] \tag{12}$$

$$\mathbf{C}(q): \quad jA\mathbf{c}^{j-1} + U\mathtt{Q}(\tau^j) = \mathbf{c}^j, \quad j \in [1,q] \tag{13}$$

$$\mathbf{D}_A(\xi): \quad j\mathbf{b}^T C^{j-1} U = \mathbf{v}^T(I - \mathcal{Q}(\tau^j)), \quad j \in [1,\xi] \tag{14}$$

$$\mathbf{D}_B(\xi): \quad j\mathbf{b}^T C^{j-1} A = \mathbf{b}^T(I - C^j), \quad j \in [1,\xi] \tag{15}$$

with $p \leq 2q+2$ and $p \leq q+\xi+1$, then the method will be of order p. Note that C and $\mathcal{Q}$ are diagonal matrices formed by the entries of $\mathbf{c}$ and $\mathtt{Q}$, respectively, and τ^j is a "bushy" tree: trees for which all subtrees are of order 1.

Note that these conditions reduce to those derived by Butcher [15] when applied to RK methods. The first condition $\mathbf{B}(p)$ (12) corresponds to the full order conditions for "bushy" trees τ^p. The second condition $\mathbf{C}(q)$ (13) is the requirement that the MRK method have minimum stage order q across all stages. Finally, the combination of the last three simplifying conditions, $\mathbf{C}(q)$ (13), $\mathbf{D}_A(\xi)$ (14) and $\mathbf{D}(\xi)_B$ (15), 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.

2.2 Linear Stability

While the governing equations in CFD are most often nonlinear, linear stability is an important necessary condition. Therefore, this article focuses on linear stability. Nonlinear stability definitions can be considered, such as algebraic stability, but these are left to future work.

A-stability: A time-marching method is said to be A-stable, if it is unconditionally stable for the linear IVP [2]

$$\mathcal{Y}' = \lambda\mathcal{Y}, \quad \mathcal{Y}(t_0) = y_0, \quad t_0 \leq t \leq t_f, \tag{16}$$

where $\lambda \in \mathbb{C}$ and $\mathrm{Re}(\lambda) \leq 0$. Specifically, an MRK method will be A-stable if the eigenvalues of the stability matrix

$$M(z) = V + zB(I - zA)^{-1}U, \tag{17}$$

for all $z = \lambda h$ in the left-half complex plane, lie within the unit disk, and any eigenvalue on the unit disk is simple.

L-stability: An extension of A-stability, which guarantees damping of stiff parasitic modes, is called L-stability [16]. An MRK method is called L-stable if it is both A-stable and the eigenvalues of the stability matrix satisfy

$$\lambda_{M(z)} \to 0 \text{ as } |\lambda| \to \infty. \tag{18}$$

A stiffly accurate MRK method with non-singular A coefficient matrix will satisfy the additional condition (18) for L-stability automatically (by extension of Proposition 3.8 in Section IV.3 of Hairer and Wanner [11]).

A(α)-stability: If an MRK method satisfies the A-stability condition in a wedge of the left-half complex plane defined by the angle α with respect to the negative real axis and rooted at the origin, then the method is said to be A(α)-stable. If the method also provides damping of parasitic modes, $\lambda_{M(z)} \to 0$ as $|\lambda| \to \infty$, the method is called L(α)-stable.

Internal A- and L-stability: The stability of the internal stages of MRK methods can be defined similarly, called internal A- and L-stability [17]. For this case, we consider the following stability matrices

$$M(z,i) = U_{i,:} + zA_{i,:}(I - zA)^{-1}U, \qquad \text{for } i = 1,\ldots,s. \tag{19}$$

2.3 Local Truncation Error

For the linear differential equation (16), the exact solution can be written as

$$\mathcal{Y}(t) = y_0 e^z = y_0 \sum_{j=0}^{\infty} \frac{z^j}{j!}. \tag{20}$$

Applying a multistep Runge-Kutta method, the solution can be updated using the stability matrix (17) defined above

$$\begin{bmatrix} y^{[n+1]} \\ y^{[n]} \\ \vdots \\ y^{[n-r+2]} \end{bmatrix} = M(z) \begin{bmatrix} y^{[n]} \\ y^{[n-1]} \\ \vdots \\ y^{[n-r+1]} \end{bmatrix}. \tag{21}$$

Each eigenvalue of the stability matrix can be expanded using Taylor series. The eigenvalue which approximates e^z is called the principal eigenvalue and defines the local truncation error (LTE) with coefficient $\mathtt{C}_{\mathrm{LTE}} \in \mathbb{R}$

$$\lambda_{M,p} = e^z + \mathcal{O}(h^{p+1}) = \sum_{j=0}^{p} \frac{z^j}{j!} + \mathtt{C}_{\mathrm{LTE}}(p+1)\frac{z^{p+1}}{(p+1)!} + \mathcal{O}(h^{p+2}). \tag{22}$$

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.

3 Numerical Optimization

The construction and numerical optimization of MRK methods is carried out using a custom package in Maple 18. The package is written for the optimization of general linear methods, of which MRK methods are a subset, and includes analysis of additional features not considered in this article, such as algebraic stability.

3.1 Design Variables

The skeleton of the method is constructed by selecting the number of steps r, the number of implicit stages s, and whether to enforce stiff accuracy, then solving the conditions for desired order p and stage order q. Selection of which coefficients are solved for first can aid the solution process. For example, abscissa values tend to appear with higher exponents in the order conditions. Therefore, they are solved for last to minimize computational effort and the introduction of multiple solutions.

Where possible, the additional condition for L-stability ($\lim_{z\to\infty} M_{\mathrm{A}}(z) = 0$) is solved for at this point as well. As the number of steps r is increased, this becomes more difficult due to the order of the resulting characteristic polynomial. In this article, we circumvent this difficulty by considering only stiffly accurate methods on the assumption that the A coefficient matrix remains invertible.

The remaining undetermined coefficients are used as design variables in the optimization.

3.2 Objective Function

We are interested in methods which balance accuracy, internal stability, conditioning, and computational cost. To optimized relative to these objectives, we apply a composite objective function. The first part of the objective function measures numerical accuracy. This is accomplished by using the L$_2$-principal error norm [18]:

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

where $\mathtt{O}(\mathtt{t})$ are the order conditions (9). This is the norm of the violation in the conditions one order higher than the method is designed for.

The internal L-stability of the method forms the second part of the objective function related to the conditioning of the method. Large values can lead to inaccurate stage solutions and poor computational convergence. The internal L-stability is considered in the objective function using

$$\mathcal{S} = \textstyle\sum_i \sum_{\forall \lambda} \mathbb{R}(\lambda_{M(\infty,i)}) \tag{24}$$

The third and fourth parts of the objective function address the conditioning and computational cost of the method through the spacing of the abscissa values, the magnitude of the diagonal entries in the A coefficient matrix, and the eigenvalues of the matrix $(I - A)^{-1}$:

$$\mathcal{C}_1 = \frac{\sqrt{([\mathbf{c}^T\ 1] - [0\ \mathbf{c}^T])\left([\mathbf{c}^T\ 1] - [0\ \mathbf{c}^T]\right)^T}}{s} + \sqrt{\sum_i A_{i,i}^2}, \quad \mathcal{C}_2 = \sqrt{\sum_i \lambda^2_{(I-A)^{-1},i}}. \tag{25}$$

Smaller spacing between the abscissa values will help generate better initial iterates for a Newton's method based solver. Smaller diagonal entries in the A coefficient matrix will make the system more diagonally-dominant, further contributing to rapid convergence of a Newton's method based solver. The second function $\mathcal{C}_2$ is another attempt to recover a well-conditioned method.

The final composite objective function looks like

$$\mathcal{J} = \chi_1 \mathcal{E}(p) + \chi_2 \mathcal{S} + \chi_3 \mathcal{C}_1 + \chi_4 \mathcal{C}_2 \tag{26}$$

where the coefficients χ_i can be manipulated to create a Parato front comparing the competing objectives. Only a few variations of the coefficients χ_i are chosen in this article. Future work will investigate the trade-offs related to a greater number of variations. The objective function is evaluated during the optimization by first substituting the design variables into the coefficient vectors and matrices, then numerically computing the different functions. This is in contrast to precomputing symbolic expressions for the functions in terms of the design variables, then substituting the design variables into these expressions. The approach taken minimizes the number of large complex symbolic expressions which need to be computed and stored.

The gradient of the objective functions is computed using finite differences with a step size of 10^{-10}. The step size is chosen to balance accuracy and round-off error (See *e.g.* [19]). Note also that the code runs with double precision. A complex step method [20] is also implemented but is not compatible with the stability components of the objective function.

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 [21].

3.3 Constraints

Linear and nonlinear constraints are applied to enforce global stability criteria and to place a bound of the coefficients of the method. A similar approach is taken to the objective function: evaluating the constraints during the optimization after first substituting the design variables into the coefficient vectors and matrices and computing the gradient of the constraints with finite differences and a step size of 10^{-10}.

A-stability: The constraint for A-stability is implemented using the concept of the stability contour applied to each eigenvalue of the stability matrix $M(z)$. The stability contours are obtained by solving $|M(z) - e^{\mathtt{i}\theta} I|$ for the associated complex z-coordinates, where $\mathtt{i} = \sqrt{-1}$ and $\theta \in [0, \pi)$. Observe that the magnitude of $e^{\mathtt{i}\theta}$ is unity, thus defining the boundary between the stable and unstable regions. We also only solve for $\theta \in [0, \pi)$ since the contours are symmetric about the real axis of the complex z-plane. This region is then discretized using typically 40 non-equidistant points clustered around the intersection of the contour and the origin. The real components of the z-coordinates are then solved for and constrained to be greater than or equal to zero such that the stability contours lie in the right-half complex z-plane. Additional constraints, $\lambda_{M(z=-1)} \leq 1$, are used to ensure that it is the stable region of the contours which lie in the left-half complex plane.

L-stability: The additional criteria for L-stability, $\lambda_{M(z)} \to 0$ as $z \to \infty$, are typically enforced before the optimization to determine the relationship between additional coefficients. Alternatively, the condition can be enforced as a constraint during the optimization.

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

$$\begin{array}{lll} \pm(A_{ij} - \Gamma) \leq 0, \quad \pm(b_i - \Gamma) \leq 0, & \text{for} & i, j = 1, \ldots, s. \\ \pm(U_{ij} - \Gamma) \leq 0, \quad \pm(v_j - \Gamma) \leq 0, & \text{for} & i = 1, \ldots, s, \quad j = 1, \ldots, r. \\ \pm(c_i - \Gamma) \leq 0, & \text{for} & i = 1, \ldots, s \end{array} \tag{27}$$

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 constraints. The design space is often multi-modal, having several local minima; therefore a multi-start procedure is used [22]. Initial values for the design variables are generated with a Sobol sequence [22, 23, 24] in a predetermined range. Typically 400 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 Fourth-Order Implicit Multistep 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 MRK methods. In the discussion below, we only present the properties of the most relevant schemes. In the derivation of some methods, multiple solutions (skeleton methods) were obtained from the order conditions. Not all solutions yielded competitive methods based on the characteristics considered in this article. These methods were discarded and are not presented here. Coefficients for the most promising new methods can be found in Appendix A.

In this article we use a similar form of identification for the time-marching methods as that used by Kennedy and Carpenter [25]:

Class[p,(q$_i$)](s,r)X_SA_i

Class	The methods classification: implicit linear multistep (ILM), singly-diagonally-implicit Runge-Kutta (SDIRK) and multistep Runge-Kutta (SDIMRK) mthods
p	Order of the method
q$_i$	Stage order of the individual stages - if the stage orders are equal for the first $s-1$ stages, only the minimum stage order is shown

s	Number of stages
r	Number of steps
X	Stability property (A,L)
SA	Stiffly accurate
i	Unique identifier

One aim of this work was to compare to known methods in the literature. However, very few stiffly accurate L-stable SDIMRK methods were found. Kerr and Burrage [26] considered such methods with the inclusion of an explicit first stage, Burrage and Chipman [27, 28] methods which are not stiffly accurate by definition. Many other investigations of MRK consider only explicit or fully implicit variants.

Efficiency is defined by the ratio between solution accuracy and computational work. The metric of accuracy defined above is the L$_2$-principal error norm (23); however, this does not take into account the relative computational cost of each method. To measure the relative efficiency of each diagonally-implicit method, the relative error norm is introduced [6]:

$$\mathcal{E}_{\mathrm{rel}}(p) = \mathcal{E}(p) s_{\mathrm{i}}^{p}, \tag{28}$$

where s_{i} is the number of implicit stages. The number of implicit stages is used here as a surrogate for computational intensity. In reality there are many factors which can influence the efficiency of a scheme, such as the spacing between abscissa values, or the magnitude of the diagonal coefficients. A similar approximation can be derived for the local truncation error coefficient:

$$\mathtt{C}_{\mathrm{LTE/rel}}(p+1) = \mathtt{C}_{\mathrm{LTE}}(p+1) s_{\mathrm{i}}^{p}. \tag{29}$$

These approximations are designed to give a first order approximation of relative efficiency; numerical simulation is required to more fully compare the schemes. This is presented in Section 5.

4.1 Single-stage MRK methods

As a starting point, consider the fourth-order backward difference formula (BDF4). This four-step implicit linear multistep method detailed in Table 1 is entirely defined by the fourth-order accuracy conditions. Relative to other fourth-order multistage methods from the literature, or optimized in this work, BDF4 has a large L$_2$-principal error norm. However, it only requires a single function evaluation per time step. Therefore, for a fixed computational cost, it can use much smaller time step sizes. Given the order of the method, the exponential reduction in error obtained by the smaller time step size far outweighs the large error norm. This makes BDF4 very efficient.

BDF4 also provides damping of stiff modes, $\rho(M(z)) \to 0$ as $z \to \infty$. However, it is does not satisfy the conditions for A-stability. BDF4 is only L(76.36°)-stable. The stability angle is relatively large, meaning that BDF4 could be applied to diffusion-dominated problems; however, it is likely to be problematic for the majority of CFD simulations. Even the third-order variant BDF3, which has a stability angle of $\alpha = 86.03°$, is not commonly applied in CFD due to the lack of full A-stability [30].

Additional steps can be added to the linear multistep approximation to increase the stability angle of the method, as shown in Table 1, or to reduce the error norm [31]. However, A-stability cannot be fully recovered for a fourth-order linear multistep methods, independent of the number of steps [2].

4.2 Two-stage MRK methods

To obtain a fourth-order method with full A- or L-stability, requires at least two implicit stages. The most efficient L-stable two-stage MRK methods optimized, SDIMRK[4,3](2,3)L_SA_3 and SDIMRK[4,3](2,4)-L_SA_2, have very similar error norms to BDF4, as shown in Table 2. On one hand, the amount of computational work required per time step relative to BDF4 is about double, increasing the relative error norm of these methods by nearly a factor of ten. On the other hand, the addition of L-stability can help generate solutions to stiff problems where BDF4 fails. This is shown in for laminar flow over a circular in Section 5.

Considering only the two-stage methods, the data shows that increasing from three to four steps improves the relative spacing of the abscissa values. This can improve the accuracy of initial iterates for a Newton-type

Method	Objective function	$p_{\text{Simp}}[p, q, \xi_A, \xi_B]$	$s\left(C_1 - \sqrt{\sum_i A_{i,i}^2}\right)$	$\min(0, \mathbf{c}) - \max(1, \mathbf{c})$	$A_{i,i}$	$\max_i M(\infty, i)$	$\mathcal{E}(p)$	$\mathcal{E}_{\text{rel}}(p)$	$\lvert\mathsf{C}_{\text{LTE/rel}}(p+1)\rvert$	Reference
ILM[4](4)L(73.36°)_SA (BDF4)	–	4 [4,4,0,0]	1.00	0.00 – 1.00	0.48	0.00	34.56	34.56	24.00	*[29]
ILM[4](5)L(76.09°)_SA	$\chi_i = 1$	4 [4,4,0,0]	1.00	0.00 – 1.00	0.51	0.00	62.22	62.22	–	*

Table 1: Implicit linear multistep methods. The asterisk denotes a method optimized or rederived using the numerical procedure described in Section 3. Symbolic computation of the local truncation error becomes increasingly difficult as the number of steps increases. Therefore, no value of $|\mathsf{C}_{\text{LTE/rel}}(p+1)|$ is shown for five-step methods.

Method	Objective function	$p_{\text{Simp}}[p, q, \xi_A, \xi_B]$	$s\left(C_1 - \sqrt{\sum_i A_{i,i}^2}\right)$	$\min(0, \mathbf{c}) - \max(1, \mathbf{c})$	$A_{i,i}$	$\max_i M(\infty, i)$	$\mathcal{E}(p)$	$\mathcal{E}_{\text{rel}}(p)$	$\lvert \mathtt{C}_{\text{LTE/rel}}(p+1)\rvert$	Reference
SDIMRK[4,3](2,3)L_SA_3	$\chi_i = 1$	4 [4,3,0,0]	1.90	$0.00 - 1.75$	0.83	0.00	35.66	570.61	387.76	*[28]
ESDIMRK[4,3](3,3)A_SA†	–	4 [4,3,0,0]	1.58	$0.00 - 1.50$	0.94	1.88	53.34	853.38	559.75	[26]
SDIMRK[4,3](2,4)L_SA_2	$\chi_i = [1, 0, 0, 0]$	4 [4,3,0,0]	1.51	$0.00 - 1.45$	0.84	0.00	36.69	587.11	462.70	*
SDIMRK[4,3](2,4)L_SA	$\chi_i = 1$	4 [4,3,0,0]	0.97	$0.00 - 1.00$	0.83	0.00	48.64	778.18	692.72	*
SDIMRK[4,3](2,5)L(84.99°)_SA	$\chi_i = [1, 0, 0, 0]$	4 [4,3,0,0]	2.64	$0.00 - 2.30$	0.78	0.00	37.47	599.47	–	*

Table 2: Two-step multistep Runge-Kutta methods. The asterisk denotes a method optimized or rederived using the numerical procedure described in Section 3. The symbol † denotes a method with an explicit first stage, meaning the number of implicit stages is one less than reported. Symbolic computation of the local truncation error becomes increasingly difficult as the number of steps increases. Therefore, no value of $\lvert \mathtt{C}_{\text{LTE/rel}}(p+1)\rvert$ is shown for five-step methods.

solver and thus reduce computational cost to some degree. In contrast, no benefit in the relative error norm was observed by increasing the number of steps from three to four or five.

In general we found that as the number of steps increased, the number of times each local minimum was found decreased. In other words, it seemed to make the optimization problem more multi-modal. The exact reason for this is not known. One possibility is the increasing size of the stability matrix, which increases the complexity of the stability conditions. Another possibility is the number of undetermined coefficients. We found that the relative increase in undetermined coefficients was greater when adding additional steps than when adding additional stages. Future work should consider improving the conditioning of the optimization procedure for multistep methods.

4.3 Three and four-stage MRK methods

Consider the three and four-stage MRK methods detailed in Table 3, specifically, those which make use of more than one step (*i.e.* not RK methods). Numerical optimization is able to reduce the error norm of these methods by one to two orders of magnitude relative to BDF4 and the two-stage methods discussed previously. Even with the additional computational cost associated with the increased number of stages, the three-stage SDIMRK[4,2](3,2)L(89.97°)_SA and four-stage SDIMRK[4,2](4,2)L_SA_0 methods are expected to be four and five times more efficient than the two-stage methods, respectively. While still expected to be less efficient than BDF4, these methods are L-stable, which can have significant impact in practical simulations.

The two reference 3-step 3-stage methods included in the table are implicit advance step point (IAS) methods. These methods prescribe a particular abscissa in order to generate a superfuture point, along with a high-order corrector on the last stage. Relaxing this prescription in the present optimizations enabled a significant increase in efficiency.

4.4 Runge-Kutta methods

Finally, we compare the results of the single step Runge-Kutta methods shown in Table 4. The most efficient of these methods is the reference method SDIRK[4,1](5)L_SA_ha followed closely by SDIRK[4,1](5)L-_SA_2, which was optimized with the present approach. The former method was rederived with the current optimization tool in a previous article [7], but using a different objective function. The trade-off between the methods is an exchange of error norm and abscissa spacing. Despite having the greatest number of stages and thus requiring the greatest computational effort per time step, both methods are predicted to be more efficient than any other L-stable multistep method considered based on relative error norm.

A general trend we observe in the data using the particular objective defined in Section 3 is that adding additional steps seems to have a negative impact on the error norm. This is in direct opposition to our expectation. The deciding factor in efficiency then comes in whether or not the the reduction in computational cost per time step or error norm is more significant. In this case, the best RK method SDIRK[4,1](5)L_SA_ha should be about 25% more efficient than best L-stable multistep method SDIMRK[4,2](4,2)L_SA_0. Future work should investigate different objective functions which may have a different impact on the results.

The potential drawback of the RK methods is lower stage order relative to the multistep methods. In general, the highest stage order a RK method can obtain is order two, which requires the inclusion of an explicit first stage (not considered in this article). This limits high-order RK methods when applied to problems which exhibit order reduction. This has been observed, for example, with Reynolds-Averaged Navier-Stokes simulations [33, 7]. In that case, MRK methods may have an advantage not captured by the predicted relative efficiency shown in the tables.

5 Numerical Simulations

Verification of the optimized MRK methods' order properties is presented using numerical simulation of van der Pol's equation [34]. This is done for both nonstiff and stiff variants. The relative efficiency of the methods is then compared using numerical simulation of laminar flow over a circular cylinder at a Reynolds number of 1200.

Method	Objective function	$p_{\text{Simp}}[p, q, \xi_A, \xi_B]$	$s\left(\mathcal{C}_1 - \sqrt{\sum_i A_{i,i}^2}\right)$	$\min(0, \mathbf{c}) - \max(1, \mathbf{c})$	$A_{i,i}$	$\max_i M(\infty, i)$	$\mathcal{E}(p)$	$\mathcal{E}_{\text{rel}}(p)$	$\mathrm{C}_{\text{LTE/rel}}(p+1)$	Reference
SDIMRK[4,2](3,2)L(89.97°)_SA	–	3 [4,2,0,0]	0.61	$0.00 - 1.00$	0.38	0.00	1.88	152.20	98.60	*[6]
SDIMRK[4,2](3,2)L_SA_0	–	3 [4,2,0,0]	2.68	$0.00 - 2.47$	1.44	0.00	151.52	12273.07	9197.65	*[6]
SDIMRK[4,2](3,3)L_SA_1	$\chi_i = [1, 0, 0, 0]$	3 [4,2,0,0]	2.86	$0.00 - 2.62$	1.09	0.00	3.23	261.69	180.63	*
SDIMRK[4,2](3,3)L_SA_3	$\chi_i = 1$	3 [4,2,0,0]	1.32	$0.00 - 1.00$	0.39	0.00	7.79	630.64	159.29	*
SDIMRK[4,2](3,3)L_SA	–	3 [4,2,0,0]	1.73	$0.00 - 2.00$	0.44	0.00	14.79	1198.18	241.29	[6]
SDIMRK[4,3](3,3)L_SA	–	4 [4,3,0,0]	1.73	$0.00 - 2.00$	0.55	0.00	20.84	1687.78	866.66	[32]
SDIMRK[4,2](3,4)L(89.97°)_SA	$\chi_i = [1, 1, 1, 0]$	3 [4,2,0,0]	2.84	$0.00 - 1.94$	0.78	0.00	20.98	1699.31	1544.52	*
SDIMRK[4,2](4,2)L_SA_0	$\chi_i = 1$	3 [4,2,0,0]	0.60	$0.00 - 1.00$	0.22	0.00	0.43	109.23	44.71	*
SDIMRK[4,2](4,2)L_SA	–	3 [4,2,0,0]	1.21	$0.00 - 1.00$	0.49	0.00	1.85	474.02	$1.54e - 7$	[6]
SDIMRK[4,3](4,3)L_SA	–	4 [4,3,0,0]	2.65	$0.00 - 3.00$	0.55	0.00	1.43	365.45	171.34	[6]
SDIMRK[4,(2,1,2,4)](4,3)L_SA_0	$\chi_i = [1, 0, 0, 0]$	2 [4,1,0,0]	6.15	$-3.71 - 1.00$	0.40	0.00	3.21	820.99	441.81	*
SDIMRK[4,(2,1,2,4)](4,3)L_SA_1	$\chi_i = [1, 1, 1, 0]$	2 [4,1,0,0]	1.12	$0.00 - 1.00$	0.38	0.00	6.32	1619.18	510.96	*

Table 3: Three and four-step multistep Runge-Kutta methods. The asterisk denotes a method optimized or rederived using the numerical procedure described in Section 3.

Method	Objective function	$p_{\text{Simp}}[p, q, \xi_A, \xi_B]$	$s\left(\mathcal{C}_1 - \sqrt{\sum_i A_{i,i}^2}\right)$	$\min(0, \mathbf{c}) - \max(1, \mathbf{c})$	$A_{i,i}$	$\max_i M(\infty, i)$	$\mathcal{E}(p)$	$\mathcal{E}_{\text{rel}}(p)$	$\mathtt{C}_{\text{LTE/rel}}(p+1)$	Reference
SDIRK[4,1](5)L_SA_ha	–	2 [4,1,4,0]	0.78	0.00 – 1.00	0.25	0.00	0.13	83.51	63.48	[11, 7]
SDIRK[4,1](5)L_SA_2	$\chi_i = 1$	2 [4,1,4,0]	0.76	0.00 – 1.00	0.25	0.00	0.13	83.85	66.31	*

Table 4: Runge-Kutta methods. The asterisk denotes a method optimized or rederived using the numerical procedure described in Section 3.

5.1 Van der Pol's Equation

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

$$\mathcal{Y}''(x) - \mu\left(1-\mathcal{Y}(x)^2\right)\mathcal{Y}'(x) + \mathcal{Y}(x) = 0, \tag{30}$$

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

$$\left\{ \begin{array}{c} \mathcal{Z}_1'(t) = \mathcal{Z}_2(t) \\ \epsilon\mathcal{Z}_2'(t) = \left(1-\mathcal{Z}_1(t)^2\right)\mathcal{Z}_2(t) - \mathcal{Z}_1(t) \end{array} \right\}, \tag{31}$$

where ϵ is called the stiffness parameter. For large values of ϵ the problem is nonstiff and the expected order of convergence is p. As ϵ is lowered, the convergence theory becomes more closely related to differential algebraic equations (DAEs). Indeed, if $\epsilon \to 0$ we recover an index-1 DAE. Schneider [35] showed that A-stable general linear methods, of which MRK methods form a subset, applied to a singular perturbation problem achieve a convergence rate of

$$\mathcal{Z}_1(t_\circ + n\,h) - z_1^{[n]} = \mathcal{O}(h^p) + \mathcal{O}(\epsilon h^{q+1}), \text{ and } \mathcal{Z}_2(t_\circ + n\,h) - z_2^{[n]} = \mathcal{O}(h^{q+1}). \tag{32}$$

Unfortunately, no result is given for the case where the general linear method is stiffly accurate. There is, however, a result for stiffly-accurate RK methods. In this case, the convergence of the second variable becomes [36]:

$$\mathcal{Z}_2(t_\circ + n\,h) - z_2^{[n]} = \mathcal{O}(h_N^p) + \mathcal{O}(\epsilon h_N^q). \tag{33}$$

For this study 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 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 [11], and the time domain is set to $t = [0, 0.5]$ to be consistent with the literature [37, 11].

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

$$e_{z_{1/2}} = \sqrt{\frac{\sum_{i=1}^{N}(z_{1/2,i} - z_{1/2,i,\text{ref}})^2}{N}}, \tag{34}$$

where N represents the number of time steps. The reference solution is computed with the ESDIRK[4,2](6)L_SA [37] method and a time step 2^{-17}. This is several orders of magnitude smaller than that used to generate the convergence results presented below. ESDIRK[4,2](6)_SA is also used to compute the first $r-1$ time steps in each simulation, as multistep methods are not self-starting. In other words we need a method to generate the initial vector of values to be passed to the multistep methods. The data shows no indication that the use of ESDIRK[4,2](6)L_SA has had any impact on the accuracy or convergence rates of the methods considered.

Table 5 summarizes the convergence rates obtained for simulations of van der Pol's equation using both the nonstiff and stiff variants. In the nonstiff, case the methods all recover their design order, as listed in Section 4. In the stiff case, the methods all converge at a minimum at the rate predicted by Schneider's theory. In some cases higher convergence rates are observed, but never higher than the design order. In general, the convergence rate will transition from the higher rate of p to the lower rate of q or $q+1$ as the time step size is reduced. Therefore, the higher rate of convergence may be a consequence of some methods not reaching the region of order reduction before hitting round-off error. Overall, these simulations verify the ability of the numerical tool presented in Section 3 to generate methods with prescribed accuracy properties.

5.2 Navier-Stokes Equations

The DIABLO flow solver solves the compressible Navier-Stokes equations using fourth-order classical finite-difference summation-by-parts (FD-SBP) operators on structured multi-block grids. Simultaneous approximation terms (SATs) are used to weakly enforce block-interface coupling and boundary conditions. The viscous fluxes are computed with the application of the first derivative twice, and matrix artificial dissipation

Scheme	r	s	p	q	p_{z_1}/p_{z_2}	
					Nonstiff ($\epsilon = 0.1$)	Stiff ($\epsilon = 10^{-5}$)
ILMM4(4)L(73.36°)_SA	1	4	4	4	3.9775 / 3.7870	3.9462 / 3.9521
ILMM4(5)A(76.09°)_SA	1	5	4	4	3.9701 / 3.8492	3.9304 / 3.9363
SDIMRK[4,3](2,3)L_SA_3	2	3	4	3	3.9935 / 3.7199	3.9954 / 4.0191
ESDIMRK[4,3](3,3)A_SA	3	3	4	3	3.9805 / 3.8864	3.9956 / 4.0542
SDIMRK[4,3](2,4)L_SA_2	2	4	4	3	3.9073 / 3.7109	4.1110 / 4.1925
SDIMRK[4,3](2,4)L_SA	2	4	4	3	3.8645 / 3.8774	3.9415 / 3.9198
SDIMRK[4,3](2,5)L(84.99°)_SA	2	5	4	3	4.0731 / 3.9010	3.9883 / 3.9739
SDIMRK[4,2](3,2)L(89.97°)_SA	3	2	4	2	3.6255 / 3.8289	4.0362 / 1.9280
SDIMRK[4,2](3,2)L_SA_0	3	2	4	2	3.8822 / 3.8125	4.0511 / 3.0179
SDIMRK[4,2](3,3)L_SA_1	3	3	4	2	4.0894 / 3.8886	4.3645 / 4.5614
SDIMRK[4,2](3,3)L_SA_3	3	3	4	2	4.0523 / 3.9302	4.0060 / 4.7685
SDIMRK[4,2](3,3)L_SA	3	3	4	2	4.0228 / 3.7434	3.9928 / 2.1623
SDIMRK[4,3](3,3)L_SA	3	3	4	3	4.0232 / 3.7684	3.9945 / 3.9943
SDIMRK[4,2](3,4)L(89.97°)_SA	3	4	4	2	4.0083 / 3.5594	4.0051 / 4.0250
SDIMRK[4,2](4,2)L_SA_0	4	2	4	2	3.9871 / 3.8730	4.0404 / 3.0905
SDIMRK[4,2](4,2)L_SA	4	2	4	2	3.9269 / 4.0203	4.0854 / 2.1085
SDIMRK[4,3](4,3)L_SA	4	3	4	3	4.6547 / 3.9109	4.5039 / 4.6267
SDIMRK[4,(2,1,2,4)](4,3)L_SA_1	4	3	4	2	3.9924 / 3.6547	4.0052 / 1.8524
SDIMRK[4,(2,1,2,4)](4,3)L_SA_0	4	3	4	2	3.5885 / 3.7158	4.0134 / 1.8966
SDIRK[4,1](5)L_SA_ha	5	1	4	1	3.8986 / 3.9261	3.5049 / 2.2324
SDIRK[4,1](5)L_SA_2	5	1	4	1	3.8983 / 3.9252	3.9699 / 2.2383

Table 5: **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 convergence rates were computed using a line of best fit through the 3 finest grid levels computed before round-off error.

is used to maintain numerical stability. An inexact Newton-Krylov algorithm is applied to the nonlinear residual equations using FGMRES to solve the linear system with a parallel approximate-Schur preconditioner. The convergence of Newton's method is accelerated through the use of restricted preconditioner updates, and relative tolerance nonlinear subiteration termination, discussed further below. For more information on the Diablo flow solver refer to [38, 39].

Delayed preconditioner updates Singly-diagonally-implicit methods are characterized by a constant diagonal coefficient. This means that the temporal component of the system Jacobian is constant. In many cases the spatial component varies slowly with respect to the time step size; therefore we can freeze the preconditioner over an entire stage or time step without affecting the convergence of the system of equations. 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 nonlinear equations are solved to a relative tolerance based on the value of the residual equations at the beginning of each stage. The idea is that the temporal integration will introduce a certain level of error and the residual equations need not be converged much below this error. The relative tolerance used to terminate the nonlinear iterations is fairly step size independent since better initial iterates will be generated as the step size is reduced. This reduces computational cost and can be done without any loss in global accuracy. For the comparative simulations presented below, a conservative relative tolerance of 10^{-8} was used, requiring between 4 and 8 Newton iterations per stage.

5.2.1 Laminar Flow Around a Circular Cylinder

Two-dimensional laminar vortex shedding in the wake of a circular cylinder is used to assess the relative efficiency of the optimized methods in the context of a flow problem. The Reynolds number of the flow is 1200, and the frestream Mach number is 0.2. The results are computed on a 28,000 cell grid with 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 the outer boundary is located 20 diameters from the surface of the

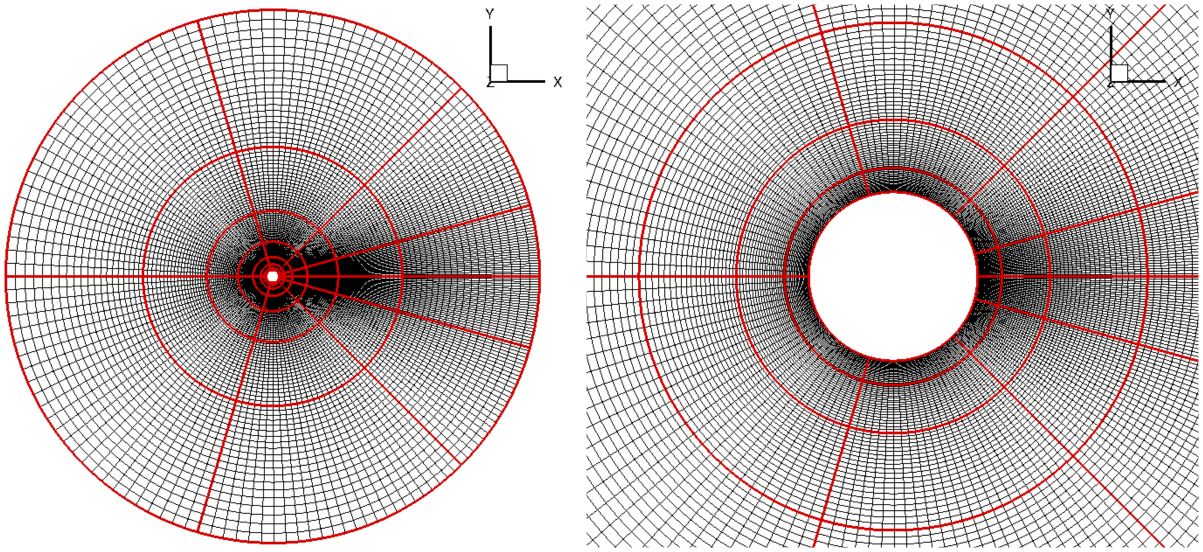

Figure 1: Computational grid with block boundaries highlighted in red

cylinder. The maximum aspect ratio is about 5.3 near the leading edge. This decreases toward the leeward side of the cylinder due to the increased circumferential resolution to capture the details of the wake. The grid is decomposed into 56 equal size blocks, and the solution is computed in parallel. Figure 1 shows the grid and block decomposition.

For these simulations, a reference solution is computed using the ESDIRK[5,2](6)L_SA method [25] and a time step size of $\Delta t = 2^{-7}$. The use of a fifth-order method in this case is in lieu of using a much finer time step size. The use of a reference solution eliminates the error associated with the spatial discretization, 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):

$$e_{C_{L/D}} = \sqrt{\frac{\sum_{i=1}^{N}(C_{L/D,i} - C_{L/D,i,\text{ref}})^2}{N}} \tag{35}$$

where N is the number of time steps.

The relative efficiency of the optimized methods is evaluated using a temporal convergence study with time step sizes ranging from $\Delta t = 2^{-7}$ to $\Delta t = 2^{3}$. The same fourth-order ESDIRK[4,2](6)L_SA method used for van der Pol's equation is applied to compute the first $r-1$ time steps for the multistep methods. The simulations are run for 80 non-dimensional time units, equal to about 5.7 vortex shedding cycles. Table 6 presents the results of the simulations. Convergence rates are computed using a line through the two smallest time steps. The total number of linear iterations required to obtain a prescribed error of $e_{C_{L/D}} = 10^{-6}$ is also shown. These values are interpolated from the data. Also shown is the average number of linear iterations per implicit stage computed from the simulations with the three smallest time steps. The methods are organised based on number of stages, followed by number of steps, and finally by predicted relative efficiency as detailed in Section 4.

The two linear multistep methods are not shown in the table since they did not converge. This is likely due to the lack of A-stability discussed previously. Apart from the linear multistep methods, the other schemes all recovered their design order for lift. The convergence rate for drag is slightly lower than the predicted values in some cases. This is due to some variations in the drag convergence plots, which occur in different locations for the different methods. Better results may have been obtained by including one or two more time levels to obtain a region with smoother convergence.

Each grouping in the table with a particular number of steps and stages follows more or less the predicted efficiency ranking listed in Section 4. This holds for both lift and drag, in spite of the lower convergence rate exhibited by some methods for drag. The one major exception is SDIMRK[4,2](3,3)L_SA_1, which

	C_L		C_D		Avg.
Method	p	Lin its ($e = 10^{-6}$)	p	Lin its ($e = 10^{-6}$)	Lin its/s_{i}
SDIMRK[4,3](2,3)L_SA_3	3.86	3.80e+05	3.19	3.65e+05	2.76e+01
ESDIMRK[4,3](3,3)A_SA	3.85	4.21e+05	3.07	4.03e+05	2.88e+01
SDIMRK[4,3](2,4)L_SA_2	3.86	3.92e+05	3.15	3.76e+05	2.77e+01
SDIMRK[4,3](2,4)L_SA	3.85	4.22e+05	3.08	4.05e+05	2.76e+01
SDIMRK[4,3](2,5)L(84.99°)_SA	4.29	3.59e+05	4.15	4.01e+05	2.72e+01
SDIMRK[4,2](3,2)L(89.97°)_SA	4.08	2.67e+05	3.82	2.59e+05	2.20e+01
SDIMRK[4,2](3,2)L_SA_0	3.70	8.61e+05	2.90	7.87e+05	3.26e+01
SDIMRK[4,2](3,3)L_SA_1	4.61	4.54e+05	3.28	5.21e+05	3.02e+01
SDIMRK[4,2](3,3)L_SA_3	4.02	2.96e+05	3.66	2.82e+05	2.22e+01
SDIMRK[4,2](3,3)L_SA	3.96	3.27e+05	3.58	3.12e+05	2.30e+01
SDIMRK[4,3](3,3)L_SA	3.87	4.41e+05	3.30	4.21e+05	2.46e+01
SDIMRK[4,2](3,4)L(89.97°)_SA	3.87	5.40e+05	3.28	5.21e+05	2.71e+01
SDIMRK[4,2](4,2)L_SA_0	3.92	2.14e+05	3.62	2.08e+05	2.31e+01
SDIMRK[4,2](4,2)L_SA	4.05	2.85e+05	3.33	3.33e+05	2.92e+01
SDIMRK[4,3](4,3)L_SA	4.15	3.62e+05	3.88	3.81e+05	2.13e+01
SDIMRK[4,(2,1,2,4)](4,3)L_SA_0	4.00	3.95e+05	3.65	3.75e+05	2.23e+01
SDIMRK[4,(2,1,2,4)](4,3)L_SA_1	4.02	3.76e+05	3.67	3.57e+05	2.19e+01
SDIRK[4,1](5)L_SA_ha	3.96	2.62e+05	3.81	2.58e+05	2.41e+01
SDIRK[4,1](5)L_SA_2	3.93	2.63e+05	3.80	2.59e+05	2.40e+01

Table 6: **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.

requires significantly more computational effort than expected. This is also reflected in the number of linear iterations per implicit stage. A relatively large diagonal coefficient, which make the solution of the nonlinear system more difficult, may be the cause of this deviation. Likewise, other methods with a large diagonal coefficient require a greater number of linear iterations per implicit stage. The spacing in abscissa values also seems to have an affect on the number of linear iterations.

While the ranking within each group is consistent with the results in Section 4, the relative magnitudes are different. In a previous paper [7], we found that the efficiency for laminar flow over a circular cylinder was more closely related to local truncation error, rather than L_2-principal error norm. The present results are consistent this observation. Accounting for the difference in number of linear iterations per stage, the relative efficiency of the methods is consistent with their relative local truncation error coefficients shown in Tables 1 through 4. For other problems, such as turbulent flow over a NACA0012 at high angle of attack, L_2-principal error norm is much more relevant [7]. For this paper we chose L_2-principal error norm as the objective function since the methods generated will be more robust across a wide range of problems.

A consequence of local truncation error being more relevant to accuracy and efficiency than the L_2-principal error norm, is that SDIMRK[4,2](4,2)L_SA_0 is the most efficient scheme for this problem, rather than the RK methods. In addition, a number of other MRK methods exhibit similar efficiency to the RK schemes as well. This highlights the need to be aware of the properties of the IVP in the optimization and selection of time-marching methods.

6 Conclusions

This article investigated the construction of fourth-order L-stable singly-diagonally-implicit multistep Runge-Kutta (MRK) methods through the use of constrained numerical optimization. Undetermined coefficients remaining after solving the desired order and stage order conditions are optimized relative to an objective function accounting for accuracy, internal stability, conditioning, and computational cost. The values are constrained by global stability properties, as well as bounds on the coefficients. The design space is found to be multi-modal, with many local minima. Therefore a quasi-random set of initial coefficients is generated using a Sobol sequence. These are then optimized in parallel.

The optimized MRK methods were compared to a linear multistep method, BDF4. This method is highly efficient but lacks A-stability. This caused the method to fail when applied to the Navier-Stokes problem considered. With the addition of one stage, L-stable MRK methods could be constructed. The methods have similar error norms relative to BDF4, but require twice as much computational effort per time step.

In contrast, increasing the number of stages to three or four yielded reductions in the error norm by one to two order of magnitude. The best optimized MRK method with four stages SDIMRK[4,2](4,2)L_SA_0 is about five times more efficient than the best two-stage method SDIMRK[4,3](2,3)L_SA_3 based on the relative L_2-principal error norm. However, the computational cost associated with those additional stages still rendered the method about three times less efficient than BDF4.

Finally, the methods were compared to Runge-Kutta methods. Despite the largest number of implicit stages, Runge-Kutta methods turned out to be more efficient than the MRK methods. They were still significantly less efficient than BDF4, but about 25% more efficient than SDIMRK[4,2](4,2)L_SA_0. The potential drawback of Runge-Kutta methods is low stage order, which can hinder performance when applied to problems exhibiting order reduction.

In general, it was found that with the selected objective function, adding additional solution points did not improve the error norm. Often adding additional steps increases the error norm. However, MRK methods are able to achieve higher-stage order than RK methods, making them an attractive alternative for stiff problems when order reduction is exhibited. Numerical solution of the Reynolds-averaged Navier-Stokes equations may be one such case.

Finally, the order properties of the optimized methods were verified for nonstiff and stiff problems with simulation of van der Pol's equations. Simulation of vortex shedding in the wake of a circular cylinder was used to evaluate the relative efficiency of the methods applied to a flow problem. The efficiency ranking of the methods for a given number of steps and stages was consistent with the predictions. The relative spacing in the abscissa affected the average number of linear iterations required per implicit stage. This caused the one observed deviation in the efficiency rankings.

The accuracy of the laminar circular cylinder simulation seems to be more related to local truncation error, rather than the L_2-principal error norm used to derived the methods. Therefore, across all methods, the SDIMRK[4,2](4,2)L_SA_0 method was the most efficient scheme for this problem. This is in contrast to the prediction that the RK schemes would be more efficient. In fact, a few other MRK methods were also competitive with the reference RK scheme.

References

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

[2] G. G. Dahlquist. A special stability problem for linear multistep methods. *BIT Numerische Mathematik*, 3(1):27–43, 1963.

[3] B. L. Ehle. High order A-stable methods for the numerical solution of systems of DEs. *BIT Numerische Mathematik*, 8:276–278, 1968.

[4] A. Guillou and J. L. Soulé. La résolution numérique des problèmes différentiels aux conditions initiales par des méthodes de collocation. *R.I.R.O.*, 3(R-3):17–44, 1969.

[5] P. D. Boom and D. W. Zingg. High-order implicit time integration for unsteady compressible fluid flow simulations. *21st AIAA Computational Fluid Dynamics Conference*, AIAA-2013-2831, 2013.

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

[7] P. D. Boom and D. W. Zingg. Optimization of high-order diagonally-implicit Runge–Kutta methods. *Journal of Computational Physics*, 371:168–191, 2018.

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

[9] S. P. Nørsett. Semi-explicit Runge-Kutta methods. Technical Report 6/74, University of Trondheim, 1974.

[10] A. Prothero and A. Robinson. On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Mathematics of Computation*, 28(125):145–162, January 1974.

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

[12] K. Burrage and P. Moss. Simplifying assumptions for the order of partitioned multivalue methods. *BIT Numerische Mathematik*, 20(4):452–465, 1980.

[13] K. Burrage and J. C. Butcher. Non-linear stability of a general class of differential equation methods. *BIT Numerische Mathematik*, 20(2):185–203, 1980.
[14] K. Burrage. Order properties of implicit multivalue methods for ordinary differential equations. *IMA Journal of Numerical Analysis*, 8(1):43–69, 1988.
[15] J. C. Butcher. Implicit Runge-Kutta processes. *Mathematics of Computation*, 18(85):50–64, 1964.
[16] 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.
[17] M. A. Kurdi. *Stable High Order Methods for Time Discretization of Stiff Differential Equations*. PhD thesis, University of California, Berkeley, December 1974.
[18] P. Prince and J. Dormand. High order embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 7(1):67–75, 1981.
[19] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, 1981.
[20] W. Squire and G. Trapp. Using complex variables to estimate derivatives of real functions. *SIAM Review*, 40(1):110–112, 1998.
[21] F. Q. Hu, M. Y. Hussaini, and J. L. Manthey. Low-dissipation and low-dispersion Runge-Kutta schemes for computational acoustics. *Journal of Computational Physics*, 124(1):177–191, March 1996.
[22] O. Chernukhin and D. W. Zingg. Multimodality and global optimization in aerodynamic design. *AIAA Journal*, 51(6):1342–1354, June 2013.
[23] S. Joe and S. J. Wright. Remarks on algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software*, 29(1):49–57, 2003.
[24] I. M. Sobol. Distribution of points in a cube and approximate evaluation of integrals. *(in Russian) Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967. (in English) U.S.S.R. Computational Mathematics and Mathematical Physics, 7 (1967), pp. 86-112.
[25] C. A. Kennedy and M. H. Carpenter. Diagonally implicit Runge-Kutta methods for ordinary differential equations. A review. Technical Report NASA/TM 2016-219173, 2016.
[26] M. Kerr and K. Burrage. New multivalue methods for differential algebraic equations. *Numerical Algorithms*, 31(1-4):193–213, 2002.
[27] K. Burrage and F. H. Chipman. Efficiently implementable multivalue methods for solving stiff ordinary differential equations. *Applied Numerical Mathematics*, 5(1-2):23–40, February 1989.
[28] K. Burrage and F. H. Chipman. Construction of A-stable diagonally implicit multivalue methods. *SIAM Journal on Numerical Analysis*, 26(2):397–413, April 1989.
[29] C. F. Curtiss and J. O. Hirschfelder. Integration of stiff equations. *Proceedings of the National Academy of Sciences of U.S.*, 38:235–243, 1952.
[30] V. N. Vatsa and M. H. Carpenter. Higher-order temporal schemes with error controllers for unsteady Navier-Stokes equations. *17th AIAA Computational Fluid Dynamics Conference*, AIAA 2005-5245(AIAA-2005-5245), June 2005.
[31] M. H. Carpenter, J. Nordström, and D. Gottlieb. Revisiting and extending interface penalties for multi-domain summation-by-parts operators. *Journal on Scientific Computing*, 45(1-3):118–150, October 2010.
[32] J. R. Cash. The integration of stiff initial value problems in ODEs using modified extended backward differentiation formulae. *Computers & Mathematics with Applications*, 9(5):645–657, 1983.
[33] M. H. Carpenter, C. A. Kennedy, H. Bijl, S. A. Viken, and V. N. Vatsa. Fourth-order Runge-Kutta schemes for fluid mechanics applications. *Journal of Scientific Computing*, 25(1):157–194, October 2005.
[34] B. van der Pol. LXXXVIII. On "relaxation-oscillations". *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):978–992, 1926.
[35] S. Schneider. Convergence results for general linear methods on singular perturbation problems. *BIT Numerische Mathematik*, 33(4):670–686, 1993.
[36] E. Hairer, C. Lubich, and M. Roche. Error of Runge-Kutta methods for stiff problems studied via differential algebraic equations. *BIT Numerische Mathematik*, 28(3):678–700, 1988.
[37] C. A. Kennedy and M. H. Carpenter. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Applied Numerical Mathematics*, 44(1-2):139–181, 2003.
[38] M. Osusky and D. W. Zingg. Parallel Newton-Krylov-Schur solver for the Navier-Stokes equations discretized using summation-by-parts operators. *AIAA Journal*, 51(12):2833–2851, December 2013.

[39] J. E. Hicken and D. W. Zingg. A parallel Newton-Krylov solver for the Euler equations discretized using simultaneous approximation terms. *AIAA Journal*, 46(11):2773–2786, November 2008.

Appendix A Coefficients of Select Optimized Multistep Runge-Kutta Methods

SDIMRK[4,2](3,3)L_SA_3

$A_{1,1} = 0.3930651483861363$
$A_{1,2} = 0$
$A_{1,3} = 0$
$A_{2,1} = -0.1148941101316164$
$A_{2,2} = 0.3930651483861363$
$A_{2,3} = 0$
$A_{3,1} = -0.1022321159502416$
$A_{3,2} = 1.183892326144308$
$A_{3,3} = 0.3930651483861363$
$b_1 = -0.1022321159502416$
$b_2 = 1.183892326144308$
$b_3 = 0.3930651483861363$
$c_1 = 0.6087294999882393$
$c_2 = 0$
$c_3 = 1$

SDIMRK[4,2](4,2)L_SA_0

$A_{1,1} = 0.2239933794528776$
$A_{1,2} = 0$
$A_{1,3} = 0$
$A_{1,4} = 0$
$A_{2,1} = 0.1202548793023635$
$A_{2,2} = 0.2239933794528776$
$A_{2,3} = 0$
$A_{2,4} = 0$
$A_{3,1} = -0.1590542957600293$
$A_{3,2} = 0.2856172586005074$
$A_{3,3} = 0.2239933794528776$
$A_{3,4} = 0$
$A_{4,1} = -0.1663155324397809$
$A_{4,2} = 1.7497161172111$
$A_{4,3} = -0.8306845264823385$
$A_{4,4} = 0.2239933794528776$
$b_1 = -0.1663155324397809$
$b_2 = 1.7497161172111$
$b_3 = -0.8306845264823385$
$b_4 = 0.2239933794528776$
$c_1 = 0.271873871931594$
$c_2 = 0.4209985678484048$
$c_3 = 0.4860560383301755$
$c_4 = 1$

SDIRK[4,1](5)L_SA_2

$A_{1,1} = 0.2479941945984302$
$A_{1,2} = 0$
$A_{1,3} = 0$
$A_{1,4} = 0$
$A_{1,5} = 0$
$A_{2,1} = 0.4826169576794777$
$A_{2,2} = 0.2479941945984302$
$A_{2,3} = 0$
$A_{2,4} = 0$
$A_{2,5} = 0$
$A_{3,1} = 0.3868393010288858$
$A_{3,2} = -0.03142363419952957$
$A_{3,3} = 0.2479941945984302$
$A_{3,4} = 0$
$A_{3,5} = 0$
$A_{4,1} = 0.2556972207268068$
$A_{4,2} = -0.075135939056669$
$A_{4,3} = 0.07002613001697444$
$A_{4,4} = 0.2479941945984302$
$A_{4,5} = 0$
$A_{5,1} = 0.9531199645442104$
$A_{5,2} = -1.72851897758253$
$A_{5,3} = 4.9316558866406$
$A_{5,4} = -3.404251068200712$
$A_{5,5} = 0.2479941945984302$
$b_1 = 0.9531199645442104$
$b_2 = -1.72851897758253$
$b_3 = 4.9316558866406$
$b_4 = -3.404251068200712$
$b_5 = 0.2479941945984302$
$c_1 = 0.2479941945984302$
$c_2 = 0.7306111522779078$
$c_3 = 0.6034098614277863$
$c_4 = 0.4985816062855425$
$c_5 = 0.9999999999999997$