Comparison of ROW, ESDIRK, and BDF2 for unsteady flows with the high-order flux reconstruction formulation

Lai Wang* and Meilin Yu†

Department of Mechanical Engineering,
University of Maryland, Baltimore County, Baltimore, MD 21250

Abstract

We conduct a comparative study of the Jacobian-free linearly implicit Rosenbrock–Wanner (ROW) methods, the explicit first stage, singly diagonally implicit Runge–Kutta (ESDIRK) methods, and the second-order backward differentiation formula (BDF2) for the high-order flux reconstruction/correction procedure via reconstruction (FR/CPR) solution of the unsteady Navier–Stokes equations. Pseudo-transient continuation is employed to solve the nonlinear equation at each stage of ESDIRK (excluding the first stage) and each step of BDF2. A Jacobian-free implementation of the restarted generalized minimal residual method (GMRES) solver is employed with a low storage element-Jacobi preconditioner to solve the linear system at each stage of ROW and each pseudo time iteration of ESDIRK and BDF2. Several numerical experiments, including both laminar and turbulent flow simulations, are conducted to carry out the comparison. We observe that the multistage ROW2 and ESDIRK2 are more efficient than the multistep BDF2, and higher-order implicit time integrators are more efficient than lower-order ones. In general, the ESDIRK method allows a larger physical time step size for unsteady flow simulation than the ROW method when the element-Jacobi preconditioner is employed, especially for wall-bounded flows; and the ROW method can be more efficient than the ESDIRK method when the time step size is refined.

Key Words

Rosenbrock–Wanner; ESDIRK; BDF2; Jacobian-Free GMRES; Flux Reconstruction/Correction Procedure via Reconstruction; High-Order Spatiotemporal Methods; Unsteady Flows

*PhD. Email: bx58858@umbc.edu

†Assistant Professor. Corresponding Author. Email: mlyu@umbc.edu

1 Introduction

High-order computational fluid dynamics (CFD) methods have been attracting much research attention in past decades due to their high-resolution and low-dissipation properties that enable high-fidelity simulation of intricate flows. High-order spatial discretization methods, such as discontinuous Galerkin (DG) methods [1, 2, 3] and FR/CPR methods [4, 5, 6, 7], have shown their capabilities of dealing with turbulent flows [8, 9, 10, 11, 12, 13, 14, 15]. Usually, the high-order explicit strong stability preserving Runge–Kutta (SSPRK) methods [16] are used to integrate the semi-discretized governing equations for unsteady flow simulation. Due to the Courant–Friedrichs–Lewy (CFL) number constraint, explicit time integration methods may not be the optimal choice for efficient numerical simulation of stiff flow problems, such as wall-bounded turbulent flows at high Reynolds numbers. To circumvent the CFL restriction, an implicit time integration method can be employed. Recent work [17, 18, 19] has confirmed that time integration methods can have significant impact on unsteady flow simulation with high-order spatial formulations. This motivates the present study to conduct a systematic comparison of accuracy and efficiency among several widely used implicit time integration methods.

One family of the implicit time integrators is the multistep method [20]. In this family, the standard BDF method is popular in the CFD community arguably due to its ease of implementation. Since a BDF method is not A-stable when the order of accuracy exceeds two, the second-order BDF2 is widely used for large-scale simulations instead of higher-order BDF methods [21, 22, 23]. One direction to search for high-order A-stable multistep methods is to "throw in additional stages, off-step points, super-future points and the like, which leads into the large field of general linear methods" [20], such as the modified extended BDF (MEBDF) [24], the split Adams–Moulton formula (SAMF) [25], and two implicit advanced step-point (TIAS) method [26, 27]. Interested readers are referred to the work by Carpenter et al. [22] on a comparative study of implicit time integrators including the fourth-order MEBDF4, and the work by Nigro et al. [28, 29] on the application of MEBDF and TIAS to Navier–Stokes equations discretized by the high-order DG method.

An alternative family of the implicit time integrators is the multistage implicit Runge–Kutta (IRK) method [30]. We note that fully coupled IRK methods are not widely used in the CFD community due to the complication of solving fully coupled nonlinear systems. Solution strategies based on the dual time stepping procedure [31] have recently been reported by Jameson [32] and several fully coupled IRK methods have been evaluated for unsteady flow simulation. To decrease computational complexity, diagonally implicit Runge–Kutta (DIRK) and singly diagonally implicit Runge–Kutta (SDIRK) methods can be used. A comprehensive review of DIRK

methods by Kennedy and Carpenter can be found in [33]. Vermeire and Vincent have analyzed the dispersion and dissipation properties of fully-discrete high-order FR methods with two SDIRK schemes, and provided insights on their suitability for implicit large eddy simulation [18]. As a special case of SDIRK, the ESDIRK method reduces the degree of the nonlinear systems of SDIRK by one. Comparisons of BDF and ESDIRK methods have been performed in [21, 23]. ESDIRK methods are found to be more efficient than BDF methods. We note that for BDF and DIRK, a nonlinear system needs to be solved at each step or each stage.

For the linearly implicit Rosenbrock method, only one linear system is to be solved at each stage. Besides, the Jacobian matrix is only evaluated once at the first stage. All these features can potentially make a Rosenbrock method more computationally efficient than a DIRK method when they have the same number of stages and order of accuracy. In a traditional Rosenbrock method, the exact Jacobian matrix [34, 35] is critical to ensure accuracy. However, in many stiff flow problems, the analytical Jacobian matrices are not easy to obtain; and the matrix-based implementation can consume tremendous memory, thus impeding efficient simulation of large-scale flow problems. The ROW method [36, 37, 38] can preserve the nominal order of accuracy with an approximate Jacobian matrix. This flexibility makes it possible to implement a Jacobian-free Krylov subspace solver. The Rosenbrock–Krylov (ROK) method [39] reformulates the Rosenbrock/ROW method such that stage vectors are obtained from the Krylov subspace using the modified Arnoldi iteration. The ROK method naturally favors the matrix-free implementation. We notice that the Rosenbrock method can suffer from order reduction for moderately stiff problems and improvements have been developed in [40]. Recent results on the performance of several Rosenbrock methods on solving Navier–Stokes equations with high-order DG-type schemes have been reported in a series of works [41, 42, 43, 44].

When the Jacobian-free implementation is used, the advantage that the Jacobian matrix of the Rosenbrock method only needs to be evaluated once for each time step does not hold anymore. In fact, the performance of ROW and ESDIRK highly depends on how the linear and nonlinear systems are solved. Blom *et al.* [45] have shown that ROW is not necessarily more efficient than ESDIRK when a second-order central finite volume method is used. Liu *et al.* [42] has conducted a comparative study of several third-order ROW methods and a third-order ESDIRK (ESDIRK3) [21] method with a third-order hierarchical WENO (weighted essentially non-oscillatory) reconstructed discontinuous Galerkin (rDG) method. They observed that the third-order ROW methods tested are more efficient than ESDIRK3. Sarshar *et al.* [46] conducted a comparative study of various matrix-free implicit time-stepping methods on solving two-dimensional (2D) Navier–Stokes equations including SDIRK, Rosenbrock, ROW, and ROK using a second-order spatial discretization. It is found that ROK can be a competitive

method compared to other implicit time integrations.

Contributions. In this work, we conduct comparison of the Jacobian-free ROW (second-, third- and fourth-order schemes), ESDIRK (second-, third- and fourth-order schemes) and BDF2 for the high-order FR solution of the unsteady compressible Navier–Stokes equations. The nonlinear systems in ESDIRK and BDF2 are solved by means of pseudo-transient continuation. The restarted GMRES solver [47] with the element-Jacobi preconditioner serves as the linear solver. The Jacobian-free implementation decreases the tremendous memory consumption of matrix-based implementation. The accuracy and efficiency of different implicit time integrators are compared using several laminar and turbulent flows, including 2D isentropic vortex propagation, 2D laminar flow over a cylinder, three-dimensional (3D) Taylor–Green vortex evolution, and the 3D transitional flow over the SD7003 wing at the chord-based Reynolds number of 60000.

Article Organization. The rest of the paper is organized as follows. The FR discretization of the 3D compressible Navier–Stokes equations is reviewed in Section 2. Section 3 introduces all the time integration methods tested in this study as well as the matrix-free implementation of the iterative methods. Numerical results and corresponding discussions are documented in Section 4. We summarize this work in the last section.

2 The flux reconstruction method

2.1 Governing equations

The 3D unsteady compressible Navier–Stokes equations can be written as

$$\frac{\partial \boldsymbol{q}}{\partial t} + \nabla \cdot \mathbb{F} = 0, \tag{1}$$

where $\boldsymbol{q} = (\rho, \rho u_j, E)^\intercal$, $j = 1, 2, 3$, is the vector of conserved variables, and $\mathbb{F}$ is the corresponding flux tensor. Specifically, ρ is the fluid density, $u_j, j = 1, 2, 3$, are the velocities in three orthogonal directions, and $E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho \sum_{k=1}^{3}(u_k u_k)$ is the total energy per unit volume, where $p = \rho R T$ is the pressure, T is the temperature, $R = c_p - c_v$ is the ideal gas constant, γ is the specific heat ratio defined as $\gamma = c_p/c_v$, and c_p and c_v are specific heat capacities at constant pressure and volume, respectively. In this study, γ is set as 1.4. The flux tensor $\mathbb{F}$ consists of the inviscid part and viscous part, which can be expressed as $\mathbb{F} = \mathbb{F}_{inv}(\boldsymbol{q}) - \mathbb{F}_{vis}(\boldsymbol{q}, \nabla \boldsymbol{q})$. Note that $\mathbb{F}_{inv}(\boldsymbol{q})$ and

$\mathbb{F}_{vis}(\boldsymbol{q}, \nabla \boldsymbol{q})$ can be rewritten in a vector format as: for any $j,\ j \in \{1,2,3\}$,

$$\mathbb{F}_{inv,j}(\boldsymbol{q}) = \begin{pmatrix} \rho u_j \\ \rho u_1 u_j + \delta_{1j} p \\ \rho u_2 u_j + \delta_{2j} p \\ \rho u_3 u_j + \delta_{3j} p \\ u_j (E + p) \end{pmatrix}, \text{ and } \mathbb{F}_{vis,j}(\boldsymbol{q}, \nabla \boldsymbol{q}) = \begin{pmatrix} 0 \\ \tau_{1j} \\ \tau_{2j} \\ \tau_{3j} \\ \sum_{k=1}^{3} u_k \tau_{kj} - K_j \end{pmatrix}, \tag{2}$$

where δ_{ij} is the Kronecker delta, τ_{ij} is the viscous stress defined as

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \mu \sum_{k=1}^{3} \frac{\partial u_k}{\partial x_k} \delta_{ij} \text{ with } i = 1, 2, 3,$$

and following Fourier's law, the heat flux K_j is defined as $K_j = -\kappa \partial T / \partial x_j$. In this study, the thermal conductivity κ is calculated from the Prandtl number Pr as $\kappa = \mu c_p / \mathrm{Pr}$. The fluid viscosity μ and the Prandtl number Pr are treated as constants.

2.2 The spatial discretization

To achieve an efficient implementation of the FR method, we transfer the Navier–Stokes equations Eq. (1) from the physical domain (x, y, z) to the computational domain (ξ, η, ζ). Thus Eq. (1) can be expressed as

$$\frac{\partial \boldsymbol{Q}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial \xi} + \frac{\partial \boldsymbol{G}}{\partial \eta} + \frac{\partial \boldsymbol{H}}{\partial \zeta} = 0, \tag{3}$$

where

$$\begin{cases} \boldsymbol{Q} = |\boldsymbol{J}| \boldsymbol{q}, \\ \boldsymbol{F} = |\boldsymbol{J}| (\boldsymbol{f} \xi_x + \boldsymbol{g} \xi_y + \boldsymbol{h} \xi_z), \\ \boldsymbol{G} = |\boldsymbol{J}| (\boldsymbol{f} \eta_x + \boldsymbol{g} \eta_y + \boldsymbol{h} \eta_z), \\ \boldsymbol{H} = |\boldsymbol{J}| (\boldsymbol{f} \zeta_x + \boldsymbol{g} \zeta_y + \boldsymbol{h} \zeta_z), \end{cases} \tag{4}$$

and

$$\boldsymbol{J} = \frac{\partial (x, y, z)}{\partial (\xi, \eta, \zeta)}, \text{ and } |\boldsymbol{J}| = \det(\boldsymbol{J}). \tag{5}$$

Herein, $\boldsymbol{f} = \mathbb{F}_1$, $\boldsymbol{g} = \mathbb{F}_2$, and $\boldsymbol{h} = \mathbb{F}_3$ are flux vectors in the x, y, and z directions of a Cartesian coordinate system. $\partial \xi_i / \partial x_j$, $i, j = 1, 2, 3$, are metrics in the coordinate transformation.

The reconstructed flux polynomial in the FR method consists of two parts. One is the local flux polynomial and the other one is the correction polynomial. On solving Eq. (3), the reconstructed polynomials $\widetilde{\boldsymbol{F}}$, $\widetilde{\boldsymbol{G}}$ and $\widetilde{\boldsymbol{H}}$ of $\boldsymbol{F}$, $\boldsymbol{G}$ and $\boldsymbol{H}$ can be expressed as

$$\begin{cases} \widetilde{\boldsymbol{F}} = \boldsymbol{F}^l + \boldsymbol{F}^c, \\ \widetilde{\boldsymbol{G}} = \boldsymbol{G}^l + \boldsymbol{G}^c, \\ \widetilde{\boldsymbol{H}} = \boldsymbol{H}^l + \boldsymbol{H}^c, \end{cases} \tag{6}$$

where the superscript 'l' stands for the local flux and 'c' stands for the correction flux. Consequently, Eq. (1) can be rewritten as

$$\frac{\partial \boldsymbol{q}}{\partial t}+\frac{\partial \boldsymbol{f}^l}{\partial x}+\frac{\partial \boldsymbol{g}^l}{\partial y}+\frac{\partial \boldsymbol{h}^l}{\partial z}+\frac{1}{|\boldsymbol{J}|}\left(\frac{\partial \boldsymbol{F}^c}{\partial \xi}+\frac{\partial \boldsymbol{G}^c}{\partial \eta}+\frac{\partial \boldsymbol{H}^c}{\partial \zeta}\right)=0. \tag{7}$$

For a hexahedral element, $\boldsymbol{F}^c$, $\boldsymbol{G}^c$ and $\boldsymbol{H}^c$ can be explicitly expressed as

$$\begin{cases} \boldsymbol{F}^c(\xi,\eta,\zeta)=(\widetilde{\boldsymbol{F}}(-1,\eta,\zeta)-\boldsymbol{F}^l(-1,\eta,\zeta))g_L(\xi) \\ \qquad\qquad +(\widetilde{\boldsymbol{F}}(1,\eta,\zeta)-\boldsymbol{F}^l(1,\eta,\zeta))g_R(\xi), \\ \boldsymbol{G}^c(\xi,\eta,\zeta)=(\widetilde{\boldsymbol{G}}(\xi,-1,\zeta)-\boldsymbol{G}^l(\xi,-1,\zeta))g_L(\eta) \\ \qquad\qquad +(\widetilde{\boldsymbol{G}}(\xi,1,\zeta)-\boldsymbol{G}^l(\xi,1,\zeta))g_R(\eta), \\ \boldsymbol{H}^c(\xi,\eta,\zeta)=(\widetilde{\boldsymbol{H}}(\xi,\eta,-1)-\boldsymbol{H}^l(\xi,\eta,-1))g_L(\zeta) \\ \qquad\qquad +(\widetilde{\boldsymbol{H}}(\xi,\eta,1)-\boldsymbol{H}^l(\xi,\eta,1))g_R(\zeta), \end{cases} \tag{8}$$

where $g_{L/R}$ are the correction polynomials. In this study, we employ the Radau polynomials to recover the nodal FR-DG method. $\widetilde{\boldsymbol{F}}$, $\widetilde{\boldsymbol{G}}$ and $\widetilde{\boldsymbol{H}}$ at the element interface are referred as numerical fluxes $\boldsymbol{F}^{num}$, $\boldsymbol{G}^{num}$ and $\boldsymbol{H}^{num}$. The inviscid common fluxes can be obtained from approximate Riemann solvers. In this study, the Roe approximate Riemann solver [48] is used to calculate the common fluxes at the cell interface in the normal directions as

$$\boldsymbol{f}^{com}_{\boldsymbol{n},inv}=\frac{\boldsymbol{f}^{+}_{\boldsymbol{n},inv}+\boldsymbol{f}^{-}_{\boldsymbol{n},inv}}{2}-\boldsymbol{R}|\boldsymbol{\Lambda}|\boldsymbol{R}^{-1}\frac{\boldsymbol{q}^{+}-\boldsymbol{q}^{-}}{2}, \tag{9}$$

where superscripts '$-$' and '$+$' denote the left and right side of the current interface, the subscript $\boldsymbol{n}=(n_x,n_y,n_z)^\intercal$ is the unit normal direction from left to right, $\boldsymbol{f}_{\boldsymbol{n}}=fn_x+gn_y+hn_z$ is the flux projection in the $\boldsymbol{n}$ direction, $\boldsymbol{\Lambda}$ is a diagonal matrix consisting of the eigenvalues of the Jacobian $\partial \boldsymbol{f}_{\boldsymbol{n}}/\partial \boldsymbol{q}$, and $\boldsymbol{R}$ consists of the corresponding right eigenvectors evaluated with the Roe-averaged variables. Numerical common fluxes can be obtained as

$$\begin{cases} \boldsymbol{F}^{num}=|\boldsymbol{J}||\nabla\xi|\boldsymbol{f}^{com}_{\boldsymbol{n}}\,\mathrm{sign}(\boldsymbol{n}\cdot\nabla\xi), \\ \boldsymbol{G}^{num}=|\boldsymbol{J}||\nabla\eta|\boldsymbol{f}^{com}_{\boldsymbol{n}}\,\mathrm{sign}(\boldsymbol{n}\cdot\nabla\eta), \\ \boldsymbol{H}^{num}=|\boldsymbol{J}||\nabla\zeta|\boldsymbol{f}^{com}_{\boldsymbol{n}}\,\mathrm{sign}(\boldsymbol{n}\cdot\nabla\zeta). \end{cases} \tag{10}$$

The common viscous fluxes are $\boldsymbol{f}^{com}_{n,vis}=\boldsymbol{f}_{vis}(\boldsymbol{q}^{+},\nabla\boldsymbol{q}^{+},\boldsymbol{q}^{-},\nabla\boldsymbol{q}^{-})$ at the cell interface. Here we need to define the common solution $\boldsymbol{q}^{com}$ and common gradient $\nabla\boldsymbol{q}^{com}$ at the cell interface. By simply taking the average of the conserved variables, we get

$$\boldsymbol{q}^{com}=\frac{\boldsymbol{q}^{+}+\boldsymbol{q}^{-}}{2}. \tag{11}$$

The common gradient is computed as

$$\nabla\boldsymbol{q}^{com}=\frac{\nabla\boldsymbol{q}^{+}+\boldsymbol{r}^{+}+\nabla\boldsymbol{q}^{-}+\boldsymbol{r}^{-}}{2}, \tag{12}$$

where $\boldsymbol{r}^+$ and $\boldsymbol{r}^-$ are the corrections to the gradients on the interface. The second approach of Bassi and Rebay (BR2) [8] is used to calculate the corrections. For the hexahedral element, the correction terms are defined as

$$\boldsymbol{r} = \gamma(\boldsymbol{q}^{com} - \boldsymbol{q}_{L/R})\boldsymbol{n}, \text{ and } \gamma = |\nabla\varpi| g'_{L/R}(\varpi)\,\mathrm{sign}(\boldsymbol{n}\cdot\nabla\varpi), \tag{13}$$

where $\varpi \in \{\xi, \eta, \zeta\}$, and $\boldsymbol{q}_{L/R}$ is the local solution on the interface. If the interface is the left boundary of the element, then the local solution $\boldsymbol{q}_L$ is used, and ϖ is -1; if the interface is the right boundary of the element, then the local solution $\boldsymbol{q}_R$ is used, and ϖ is 1. In this study, $g(\pm 1) = \pm(P+1)(P+2)/2$ is used to stabilize the FR method [49], where P is the polynomial degree of the solution.

3 Implicit time integrators and iterative methods

3.1 Implicit time integrators

We rewrite Eq. (7) in a semi-discretized form as

$$\frac{\partial \boldsymbol{q}}{\partial t} = \boldsymbol{R}(\boldsymbol{q}), \tag{14}$$

for convenience of notation. We use the method of lines approach to solve this equation: the partial differential equation is first discretized in space with the FR formulation in Section 2, and then is marched in time. In this study, three different types of implicit or linearly implicit time integration methods are studied.

The BDF Method. The first one is the backward differentiation formula. A general formulation of the s-th order BDF method for solving Eq. (14) can be expressed as

$$\boldsymbol{q}^{n+1} = \Delta t\,\omega\,\boldsymbol{R}(\boldsymbol{q}^{n+1}) + \sum_{j=1}^{s} a_j\,\boldsymbol{q}^{n+1-j}, \tag{15}$$

A BDF method is not A-stable when the order of accuracy exceeds two. In this study, we only consider the BDF2 method for comparison. For BDF2, $s = 2$, $\omega = 2/3$, $a_1 = 4/3$ and $a_2 = -1/3$. BDF2 is not able to start by itself. Therefore, at the first time step the backward Euler method or BDF1 is used.

The ESDIRK Method. The second one is the explicit first stage, single diagonally implicit Runge–Kutta method, which reads

$$\begin{cases} \boldsymbol{q}^{n+1} = \boldsymbol{q}^n + \Delta t \sum_{i=1}^{s} b_i\,\boldsymbol{R}(\boldsymbol{q}_i), \\ \boldsymbol{q}_i = \boldsymbol{q}^n + \Delta t \sum_{j=1}^{i} a_{ij}\,\boldsymbol{R}(\boldsymbol{q}_j),\ i = 1, \ldots, s, \end{cases} \tag{16}$$

where s is the number of stages. The second-order, three-stage ESDIRK2 [33], third-order, four-stage ESDIRK3 [21] and fourth-order, six-stage ESDIRK4 [21]

methods are studied in this paper. All the ESDIRK methods investigated in this study have the feature that

$$a_{ii} = \begin{cases} 0, & i = 1, \\ \omega, & i \neq 1. \end{cases} \tag{17}$$

Therefore, we can rewrite Eq. (16) as

$$\begin{cases} \boldsymbol{q}^{n+1} = \boldsymbol{q}^n + \Delta t \sum_{i=1}^{s} b_i \boldsymbol{R}(\boldsymbol{q}_i), \\ \boldsymbol{q}_1 = \boldsymbol{q}^n, \\ \boldsymbol{q}_i = \Delta t\, \omega\, \boldsymbol{R}(\boldsymbol{q}_i) + \boldsymbol{q}^n + \Delta t \sum_{j=1}^{i-1} a_{ij} \boldsymbol{R}(\boldsymbol{q}_j),\ i = 2, \ldots, s. \end{cases} \tag{18}$$

The ROW Method. The last one is the linearly implicit Rosenbrock–Wanner method. The general form of a s-stage Rosenbrock method can be written as [20]

$$\begin{cases} \boldsymbol{q}^{n+1} = \boldsymbol{q}^n + \sum_{j=1}^{s} b_j \boldsymbol{K}_j, \\ \boldsymbol{K}_i = \Delta t\, \boldsymbol{R}\left(\boldsymbol{q}^n + \sum_{j=1}^{i-1} \alpha_{ij} \boldsymbol{Y}_j\right) + \Delta t\, \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{q}} \sum_{j=1}^{i} \gamma_{ij} \boldsymbol{K}_j,\ i = 1, 2, \ldots, s. \end{cases} \tag{19}$$

The above equations are reorganized to avoid the matrix-vector product of the Jacobian matrix and the summation of stage vectors [20, 41], following

$$\begin{cases} \boldsymbol{q}^{n+1} = \boldsymbol{q}^n + \sum_{j=1}^{s} m_j \boldsymbol{Y}_j, \\ \left(\frac{\boldsymbol{I}}{\omega \Delta t} - \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{q}}\right)^n \boldsymbol{Y}_i = \boldsymbol{R}\left(\boldsymbol{q}^n + \sum_{j=1}^{i-1} a_{ij} \boldsymbol{Y}_j\right) + \frac{1}{\Delta t} \sum_{j=1}^{i-1} c_{ij} \boldsymbol{Y}_j,\ i = 1, 2, \ldots, s. \end{cases} \tag{20}$$

Similarly, $\omega = \gamma_{ii}$ is defined for the Rosenbrock method. Herein, $\boldsymbol{K}_i = \frac{1}{\omega} \boldsymbol{Y}_i - \sum_{j=1}^{i-1} c_{ij} \boldsymbol{Y}_j$, $(m_1, \ldots, m_s) = (b_1, \ldots, b_s) \boldsymbol{\Gamma}^{-1}$, $(a_{ij})_{i,j=1}^{s} = (\alpha_{ij})_{i,j=1}^{s} \boldsymbol{\Gamma}^{-1}$, $(c_{ij})_{i,j=1}^{s} = \frac{1}{\omega} \boldsymbol{I} - \boldsymbol{\Gamma}^{-1}$ and $\boldsymbol{\Gamma} = (\gamma_{ij})_{i,j=1}^{s}$ [20, 41]. The Rosenbrock–Wanner method has the flexibility of evaluating the Jacobian matrix approximately while preserving the accuracy. In this study, we restrict our attention to three popular Rosenbrock–Wanner methods, namely, the second-order, three-stage ROS2PR [37], the third-order, four-stage ROS34PW2 [36], and the fourth-order, six-stage RODASP [38]. We note that it has been shown in [40] that ROS34PW2 and RODASP can suffer from order reduction. For simplicity, we use ROW2, ROW3 and ROW4 to denote them, respectively.

The coefficients of all ESDIRK and ROW methods can be found in the Appendix.

3.2 Iterative methods

For BDF2 and ESDIRK, one needs to solve the nonlinear equations in Eq. (15) and Eq. (18). We can rewrite them as

$$\boldsymbol{F}(\boldsymbol{q}^*) = 0, \tag{21}$$

where $\boldsymbol{q}^*$ is $\boldsymbol{q}^{n+1}$ for BDF and $\boldsymbol{q}_i$ for ESDIRK, respectively. For BDF2, $\boldsymbol{F}(\boldsymbol{q}^*)$ reads

$$\boldsymbol{F}(\boldsymbol{q}^{n+1}) = \left(\frac{1}{\omega\,\Delta t}\boldsymbol{q}^{n+1} - \boldsymbol{R}(\boldsymbol{q}^{n+1})\right) - \frac{1}{\omega\,\Delta t}\sum_{j=1}^{2} a_j \boldsymbol{q}^{n+1-j}. \tag{22}$$

For ESDIRK methods,

$$\boldsymbol{F}(\boldsymbol{q}_i) = \left(\frac{1}{\omega\,\Delta t}\boldsymbol{q}_i - \boldsymbol{R}(\boldsymbol{q}_i)\right) - \frac{1}{\omega\,\Delta t}\left(\boldsymbol{q}^n + \Delta t\sum_{j=1}^{i-1} a_{ij}\boldsymbol{R}(\boldsymbol{q}_j)\right), \tag{23}$$

where $i = 2, \ldots, s$. In this work, the pseudo-transient continuation is employed [31, 21, 32]. The pseudo-transient continuation is an inexact Newton's method to solve the steady state equation $\boldsymbol{F}(\boldsymbol{q}^*) = 0$ iteratively as

$$\frac{\boldsymbol{q}^{k+1,*} - \boldsymbol{q}^{k,*}}{\Delta\tau} = -\boldsymbol{F}(\boldsymbol{q}^{k+1,*}). \tag{24}$$

Herein, k is the iteration step for the pseudo-transient continuation. Eq. (24) can be linearized as

$$\left(\frac{\boldsymbol{I}}{\Delta\tau} + \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{q}}\right)^k \Delta\boldsymbol{q}^{k,*} = -\boldsymbol{F}(\boldsymbol{q}^{k,*}). \tag{25}$$

For a steady problem, as $k \to \infty$, $\tau \to \infty$ and $\Delta\boldsymbol{q}^k \to 0$. Therefore, $\boldsymbol{F}(\boldsymbol{q}^{k,*}) \to \boldsymbol{F}(\boldsymbol{q}^*)$. The Jacobian matrix for fully implicit methods can be expressed as

$$\left(\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{q}}\right)^k = \left(\frac{\boldsymbol{I}}{\omega\Delta t} - \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{q}}\right)^k. \tag{26}$$

Substitute the Jacobian matrix into Eq. (25) to obtain the final form of the linear system in the pseudo-transient continuation procedure as follows

$$\left(\frac{\boldsymbol{I}}{\Delta\tau} + \frac{\boldsymbol{I}}{\omega\Delta t} - \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{q}}\right)^k \Delta\boldsymbol{q}^{k,*} = -\boldsymbol{F}(\boldsymbol{q}^{k,*}). \tag{27}$$

The pseudo-transient continuation procedure requires an adaptation algorithm of the pseudo time step size to complete the method. In this study, we employ the successive evolution relaxation (SER) algorithm [50] as

$$\tau^0 = \tau_{init},\ \Delta\tau^{k+1} = \min\left(\Delta\tau^k \frac{||\boldsymbol{F}||_{L_2}^{k-1}}{||\boldsymbol{F}||_{L_2}^{k}}, \Delta\tau_{max}\right). \tag{28}$$

As the pseudo time marches forward, a series of linear equations (27) is successively solved until convergence. Ideally, we would expect that when $\Delta\tau$ approaches $\Delta\tau_{max} = \infty$, the residual of the pseudo-transient procedure

will gradually converge to machine zero. However, in our practice of simulating wall-bounded flows with elements clustered in near wall regions, we have to choose a moderately large $\Delta\tau_{max}$ to ensure that the residual of the linear solver can at least drop by one order of magnitude when $\Delta\tau$ equals to $\Delta\tau_{max}$. Once the linear solver fails, we would reject the current pseudo-transient continuation iteration and decrease $\Delta\tau_{max}$ by half to continue. In this study, if not specifically mentioned, $\Delta\tau_{init} = \Delta t$ and $\Delta\tau_{max} = 10^{20}$.

Restarted GMRES is employed to solve Eq. (27) as well as the linear system in Eq.(20). All linear systems can be expressed as

$$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{b}, \tag{29}$$

where

$$\boldsymbol{A} = \boldsymbol{D}(\Delta t, \Delta\tau) - \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{q}}. \tag{30}$$

The first term $\boldsymbol{D}(\Delta t, \Delta\tau)$ on the right-hand side of the above equation is a diagonal matrix related to Δt and $\Delta\tau$. The GMRES method approximates the exact solution by a vector $\boldsymbol{x}_n \in \boldsymbol{K}_n$ that minimizes the Euclidean norm $||\boldsymbol{A}\boldsymbol{x}_n - \boldsymbol{b}||$ where $\boldsymbol{K}_n$ is the n-th Krylov subspace

$$\boldsymbol{K}_n = \text{span}\{\boldsymbol{b}, \boldsymbol{A}\boldsymbol{b}, \boldsymbol{A}^2\boldsymbol{b}, \cdots, \boldsymbol{A}^{n-1}\boldsymbol{b}\}. \tag{31}$$

The sparse matrix $\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{q}}$ in $\boldsymbol{A}$ only appears in the matrix-vector product. For an unknown vector $\boldsymbol{X}$, the matrix-vector product can be approximated as

$$\left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{q}}\right)\boldsymbol{X} = \frac{\boldsymbol{R}(\boldsymbol{q} + \varepsilon\boldsymbol{X}) - \boldsymbol{R}(\boldsymbol{q})}{\varepsilon} + O(\varepsilon). \tag{32}$$

$\varepsilon = 10^{-8}$ is employed in this study. Interested readers are referred to [51] for more discussions on this approximation. We take advantage of the framework of the restarted GMRES solver in PETSc [52] with user-defined functions to perform the matrix-vector product approximation and left preconditioning.

The performance of the Newton–Krylov method substantially depends on the preconditioner. In the context of the Jacobian-free implementation, the element-Jacobi preconditioner is arguably the simplest one with acceptable performance for many applications among the low-storage preconditioners, such as the matrix-free LU-SGS (Lower-Upper Symmetric-Gauss-Seidel) preconditioner [53], and p-multigrid preconditioner [54]. In this study, only the element-Jacobi preconditioner is considered and it is updated at the starting stage of each physical time step. The restart number of the GMRES solver is set as 60. The maximum number of iterations for GMRES are specifically designated for different numerical experiments in this work.

4 Numerical results

All the simulations presented in this section are performed on the High Performance Computing Facility (HPCF) of the University of Maryland, Baltimore county (UMBC). All nodes used in this work have two Intel E5-2650v2 Ivy Bridge (2.6 GHz, 20 MB cache) processors with eight cores apiece, for a total of 16 cores per node. A quad data rate (QDR) Infiniband switch connects all the nodes. Ideally the system can achieve a latency of 1.2 μsec to transfer a message between two nodes, and can support a bandwidth of up to 40 gigabits per second (40Gbps). Every node possesses 64 GB RAM. All nodes are running Red Hat Enterprise Linux 6.4. We employ g++ (GCC) 4.8.4 with mpich-3.1.4 to compile the code for parallel simulation.

4.1 Vortex propagation

The 2D isentropic vortex propagation problem is employed as a benchmark to investigate the accuracy and efficiency of different time integrators in this subsection. The free stream flow conditions are set as $(\rho, u, v, \text{Ma})^\intercal = (1, 1, 1, 0.5)^\intercal$ and the ideal gas constant R is set as 1.0 for this case. The perturbation is defined as [41]

$$\begin{cases} \delta u = -\frac{\alpha}{2\pi}(y - y_0)e^{\phi(1-r^2)}, \\ \delta v = \frac{\alpha}{2\pi}(x - x_0)e^{\phi(1-r^2)}, \\ \delta T = -\frac{\alpha^2(\gamma-1)}{16\phi\gamma\pi^2}e^{2\phi(1-r^2)}, \end{cases} \tag{33}$$

where $\phi = \frac{1}{2}$ and $\alpha = 5$ are parameters that define the vortex strength, and $r = (x - x_0)^2 + (y - y_0)^2$ is the distance to the center of the vortex $(x_0, y_0) = (0, 0)$ at $t = 0$. The periodic domain is defined as $\Omega = [-10, 10]^2$. We use 16 CPU processes to simulate this problem on a uniform mesh of 50×50 elements with the P^6 (i.e. $7th$ order) FR method for one period $t_p = 20$. The time step size Δt is refined from $t_p/100$ to $t_p/800$. The error of any variable s is defined as

$$Error(s) = \sqrt{\frac{\int_\Omega (s_{exact} - s_{num})^2 dV}{V}}, \tag{34}$$

where s_{exact} is the exact value, s_{num} is the numerical value, and V is the volume of the domain Ω.

Two convergence criteria are discussed in this section. The first one is the tolerance for the relative residual of the pseudo-transient continuation $tol_{rel,nonlinear}$ and the second one is that for the relative residual of the GMRES solver $tol_{rel,linear}$. Herein, the subscript 'rel' stands for 'relative'. On solving Eq. (21), $tol_{rel,nonlinear}$ will determine the error level of the pseudo-transient continuation, which will affect the order of accuracy of ESDIRK

and BDF2 and $tol_{rel,linear}$ is more related to the efficiency and robustness of the pseudo-transient continuation [55]. On solving Eq. (29), $tol_{rel,linear}$ will determine the order of accuracy of ROW directly. The maximum number of iterations of the GMRES solver is 200, which is sufficiently large for the GMRES solver to converge to the designated tolerances for this problem.

4.1.1 Effect of the nonlinear and linear convergence criteria on ESDIRK and BDF2

The pseudo-transient continuation does not require an exact solution of the linear system at each iteration. We first employ a relatively large tolerance $tol_{rel,linear} = 10^{-1}$ for GMRES to investigate the effect of $tol_{rel,nonlinear}$ on accuracy and efficiency of fully implicit methods. The convergence study of ESDIRK and BDF2 using different $tol_{rel,nonlinear}$, namely, 10^{-2}, 10^{-4}, 10^{-6} and 10^{-8}, is presented in Table 1. We observe that when $tol_{rel,nonlinear}$ is refined from 10^{-2} to 10^{-4}, all ESDIRK methods studied here will preserve the nominal order of accuracy except that ESDIRK4 shows slightly order reduction when Δt is refined from $t_p/400$ to $t_p/800$ (due to insufficient $tol_{rel,nonlinear}$); when $tol_{rel,nonlinear}$ is refined from 10^{-4} to 10^{-6}, no order reduction shows up. We also note that overrefined $tol_{rel,nonlinear}$, such as 10^{-8}, will not improve simulation accuracy, but only increase the computational cost (i.e. run time; in this study, run time is the wall clock time spent by parallel simulations). ESDIRK methods have shown better accuracy and efficiency than BDF2. Generally, the higher the order of accuracy of ESDIRK is, the more efficient ESDIRK is (see Figure 1(b)).

The impact of $tol_{rel,linear}$ on efficiency is investigated when $tol_{rel,nonlinear} = 10^{-6}$ is employed. $tol_{rel,linear}$ spans $\{10^{-1}, 10^{-2}, 10^{-4}, 10^{-6}\}$. The run time results are presented in Table 2. For this specific problem, all values of $tol_{rel,linear}$ lead to the same numerical errors as expected since the accuracy of ESDIRK and BDF2 solely depends on the convergence of pseudo-transient continuation. As documented in Table 2, the computational cost will keep on increasing when we refine the tolerance $tol_{rel,linear}$. This indicates that in order to save computational cost, $tol_{rel,linear}$ for ESDIRK and BDF2 can be relaxed.

Table 1: The convergence study for ESDIRK and BDF2 with different $tol_{rel,nonlinear}$ when $tol_{rel,linear}$ is set to 10^{-1}.

$tol_{rel,nonlinear} = 10^{-2}$, $tol_{rel,linear} = 10^{-1}$						
	$\Delta t/t_p$	$E_{L_2}(\rho)$	order	$E_{L_2}(u)$	order	Run time (seconds)
ESDIRK2	1/100	4.4097×10^{-4}		5.7334×10^{-3}		317
	1/200	1.1313×10^{-4}	1.97	1.4622×10^{-3}	1.97	603
	1/400	2.8315×10^{-5}	2.00	3.6523×10^{-4}	2.00	1159
	1/800	7.0714×10^{-6}	2.00	9.1104×10^{-5}	2.00	2247
ESDIRK3	1/100	1.5341×10^{-4}		1.7108×10^{-3}		398
	1/200	2.3518×10^{-5}	2.71	2.6017×10^{-4}	2.90	734
	1/400	3.1802×10^{-6}	2.89	3.4016×10^{-5}	2.94	1392
	1/800	1.0661×10^{-6}	1.58	9.7414×10^{-6}	1.80	2478
ESDIRK4	1/100	4.5157×10^{-6}		3.2613×10^{-5}		425
	1/200	7.7894×10^{-7}	2.54	4.7845×10^{-5}	2.77	773
	1/400	6.0568×10^{-8}	3.68	3.2420×10^{-7}	3.88	1511
	1/800	2.2122×10^{-8}	1.45	1.5485×10^{-7}	1.07	2802
BDF2	1/100	1.9170×10^{-3}		3.0021×10^{-2}		292
	1/200	8.1561×10^{-3}	1.23	1.1195×10^{-2}	1.42	548
	1/400	2.1372×10^{-4}	1.93	2.8983×10^{-3}	1.95	1074
	1/800	5.6824×10^{-5}	1.91	7.3097×10^{-4}	1.99	2133
$tol_{rel,nonlinear} = 10^{-4}$, $tol_{rel,linear} = 10^{-1}$						
	$\Delta t/t_p$	$E_{L_2}(\rho)$	order	$E_{L_2}(u)$	order	Run time (seconds)
ESDIRK2	1/100	4.4088×10^{-4}		5.7335×10^{-3}		375
	1/200	1.1278×10^{-4}	1.97	1.4549×10^{-3}	1.98	723
	1/400	2.8267×10^{-5}	2.00	3.6420×10^{-4}	2.00	1323
	1/800	7.0691×10^{-6}	2.00	9.1040×10^{-5}	2.00	2504
ESDIRK3	1/100	1.5167×10^{-4}		1.6911×10^{-3}		537
	1/200	2.3024×10^{-5}	2.72	2.5401×10^{-4}	2.74	956
	1/400	3.0921×10^{-6}	2.90	3.3171×10^{-5}	2.94	1685
	1/800	3.9376×10^{-7}	2.97	4.1908×10^{-6}	2.98	2890
ESDIRK4	1/100	3.4484×10^{-6}		3.1216×10^{-5}		597
	1/200	2.1713×10^{-7}	3.99	1.9593×10^{-6}	4.00	1046
	1/400	1.3573×10^{-8}	4.00	1.2257×10^{-7}	4.00	1920
	1/800	1.1691×10^{-9}	3.54	7.6729×10^{-9}	3.81	3193
BDF2	1/100	1.9126×10^{-3}		3.0095×10^{-2}		329
	1/200	8.4137×10^{-4}	1.18	1.1454×10^{-2}	1.39	627
	1/400	2.3001×10^{-4}	1.87	2.9892×10^{-3}	1.94	1179
	1/800	5.8417×10^{-5}	1.98	7.5360×10^{-4}	1.99	2294

To be continued on next page.

Continuation of Table 1,

$tol_{rel,nonlinear} = 10^{-6}$, $tol_{rel,linear} = 10^{-1}$

	$\Delta t/t_p$	$E_{L_2}(\rho)$	order	$E_{L_2}(u)$	order	Run time (seconds)
ESDIRK2	1/100	4.4088×10^{-4}		5.7335×10^{-3}		464
	1/200	1.1278×10^{-4}	1.97	1.4549×10^{-3}	1.98	814
	1/400	2.8267×10^{-5}	2.00	3.6420×10^{-4}	2.00	1503
	1/800	7.0691×10^{-6}	2.00	9.1041×10^{-5}	2.00	2672
ESDIRK3	1/100	1.5165×10^{-4}		1.6911×10^{-3}		692
	1/200	2.3021×10^{-5}	2.72	2.5399×10^{-4}	2.74	1138
	1/400	3.0914×10^{-6}	2.90	3.3165×10^{-5}	2.94	1877
	1/800	3.9326×10^{-7}	2.97	4.1867×10^{-5}	2.99	3369
ESDIRK4	1/100	3.4502×10^{-6}		3.1249×10^{-5}		780
	1/200	2.1706×10^{-7}	3.99	1.9593×10^{-6}	4.00	1310
	1/400	1.3584×10^{-8}	4.00	1.2270×10^{-7}	4.00	2156
	1/800	8.4961×10^{-10}	4.00	7.6728×10^{-9}	4.00	3751
BDF2	1/100	1.9126×10^{-3}		3.0096×10^{-2}		389
	1/200	8.4141×10^{-4}	1.18	1.1454×10^{-2}	1.39	708
	1/400	2.3010×10^{-4}	1.87	2.9890×10^{-3}	1.94	1230
	1/800	5.8409×10^{-5}	1.98	7.5350×10^{-3}	1.99	2463

$tol_{rel,nonlinear} = 10^{-8}$, $tol_{rel,linear} = 10^{-1}$

	$\Delta t/t_p$	$E_{L_2}(\rho)$	order	$E_{L_2}(u)$	order	Run time (seconds)
ESDIRK2	1/100	4.4088×10^{-4}		5.7335×10^{-3}		546
	1/200	1.1278×10^{-4}	1.97	1.4549×10^{-3}	1.98	930
	1/400	2.8267×10^{-5}	2.00	3.6420×10^{-4}	2.00	1601
	1/800	7.0691×10^{-6}	2.00	9.1041×10^{-5}	2.00	3003
ESDIRK3	1/100	1.5165×10^{-4}		1.6911×10^{-3}		891
	1/200	2.3021×10^{-5}	2.72	2.5399×10^{-4}	2.74	1373
	1/400	3.0913×10^{-6}	2.90	3.3164×10^{-5}	2.94	2138
	1/800	3.9326×10^{-7}	2.97	4.1867×10^{-5}	2.99	3867
ESDIRK4	1/100	3.4502×10^{-6}		3.1249×10^{-5}		971
	1/200	2.1706×10^{-7}	3.99	1.9593×10^{-6}	4.00	1550
	1/400	1.3584×10^{-8}	4.00	1.2270×10^{-7}	4.00	2543
	1/800	8.4976×10^{-10}	4.00	7.6747×10^{-9}	4.00	4370
BDF2	1/100	1.9126×10^{-3}		3.0096×10^{-2}		455
	1/200	8.4141×10^{-4}	1.18	1.1454×10^{-2}	1.39	786
	1/400	2.3010×10^{-4}	1.87	2.9890×10^{-3}	1.94	1393
	1/800	5.8409×10^{-5}	1.98	7.5350×10^{-3}	1.99	2651

End of Table 1.

Table 2: The effect of $tol_{rel,linear}$ on computational cost of ESDIRK.

		Run time (seconds)			
	$tol_{rel,linear}$	10^{-1}	10^{-2}	10^{-4}	10^{-6}
ESDIRK2	$t_p/100$	464	503	693	1041
	$t_p/200$	814	856	1109	1551
	$t_p/400$	1503	1563	1909	2501
	$t_p/800$	2672	2875	3334	4203
ESDIRK3	$t_p/100$	692	686	1036	1041
	$t_p/200$	1138	1128	1584	2419
	$t_p/400$	1877	1865	2580	3695
	$t_p/800$	3369	3526	4430	5908
ESDIRK4	$t_p/100$	780	875	1326	2095
	$t_p/200$	1310	1410	2133	3004
	$t_p/400$	2156	2439	3277	4844
	$t_p/800$	3751	4136	5527	7636

4.1.2 Effect of the GMRES convergence criterion on ROW

A comparison of different convergence criteria $tol_{rel,linear}$ of the restarted GMRES solver is conducted to study its impact on the accuracy and efficiency of the ROW time integrators. The results are presented in Table 3. It is observed that when the convergence criterion is not tight enough, such as $tol_{rel,linear} = 10^{-2}$ and 10^{-4}, the ROW methods cannot preserve the nominal order of accuracy. This is not a surprise considering that the residual convergence is directly related to the accuracy of the solution in ROW. When $tol_{rel,linear}$ is sufficiently small, such as $tol_{rel,linear} = 10^{-6}$ and 10^{-8}, all ROW methods can preserve the nominal order of accuracy, excepted that ROW4 shows order reduction [40] when the time step is refined from $t_p/400$ to $t_p/800$. We also notice that when $tol_{rel,linear}$ is refined from 10^{-6} to 10^{-8}, no significant differences in errors are observed. Another observation is that when $tol_{rel,linear}$ is overrefined, the run time of the simulation is noticeably increased. In general, we do not recommend machine zero convergence criterion for $tol_{rel,linear}$. A relatively tight value such as 10^{-6} is sufficient to preserve the accuracy of high-order Rosenbrock methods for this problem.

Table 3: The convergence study for ROW methods with different $tol_{rel,linear}$.

$tol_{rel,linear} = 10^{-2}$						
	$\Delta t/t_p$	$E_{L_2}(\rho)$	order	$E_{L_2}(u)$	order	Run time (seconds)
ROW2	1/100	Diverged	–	–	–	
	1/200	2.3615×10^{-4}		3.3198×10^{-3}		537
	1/400	9.9393×10^{-5}	1.25	1.3753×10^{-3}	1.27	1052
	1/800	5.5248×10^{-5}	0.85	7.4627×10^{-4}	0.88	2073
ROW3	1/100	2.6132×10^{-4}		2.6888×10^{-3}		300
	1/200	8.8493×10^{-5}	1.56	8.3870×10^{-4}	1.68	564
	1/400	3.3446×10^{-5}	1.40	4.0554×10^{-4}	1.05	1124
	1/800	9.1969×10^{-6}	1.86	1.1447×10^{-4}	1.82	2191
ROW4	1/100	3.6161×10^{-5}		4.1926×10^{-4}		344
	1/200	9.0325×10^{-6}	2.00	8.3350×10^{-5}	2.33	638
	1/400	2.7913×10^{-6}	1.69	2.7282×10^{-5}	1.61	1194
	1/800	3.9397×10^{-6}	-0.50	5.1692×10^{-5}	-0.92	2282
$tol_{rel,linear} = 10^{-4}$						
	$\Delta t/t_p$	$E_{L_2}(\rho)$	order	$E_{L_2}(u)$	order	Run time (seconds)
ROW2	1/100	Diverged	–	–	–	
	1/200	8.6277×10^{-5}		1.1163×10^{-3}		641
	1/400	2.1800×10^{-5}	1.98	2.8024×10^{-3}	1.99	1189
	1/800	5.4708×10^{-6}	1.99	7.1383×10^{-5}	1.97	2255
ROW3	1/100	1.5467×10^{-4}		1.7185×10^{-3}		449
	1/200	2.3151×10^{-5}	2.74	2.5539×10^{-4}	2.75	779
	1/400	3.1137×10^{-6}	2.89	3.3395×10^{-5}	2.94	1382
	1/800	4.0729×10^{-7}	2.93	4.2406×10^{-6}	2.98	2565
ROW4	1/100	4.0016×10^{-6}		3.8660×10^{-5}		487
	1/200	3.5565×10^{-7}	3.49	3.4725×10^{-6}	3.47	846
	1/400	2.7298×10^{-8}	3.70	1.6676×10^{-7}	4.38	1563
	1/800	1.8642×10^{-8}	0.55	2.2003×10^{-7}	-0.40	2769

To be continued on next page.

Continuation of Table 3,

$tol_{rel,linear} = 10^{-6}$						
	$\Delta t/t_p$	$E_{L_2}(\rho)$	order	$E_{L_2}(u)$	order	Run time (seconds)
ROW2	1/100	Diverged	–	–	–	
	1/200	8.6820×10^{-5}		1.1229×10^{-3}		767
	1/400	2.1813×10^{-5}	1.99	2.8148×10^{-4}	2.00	1377
	1/800	5.4628×10^{-6}	2.00	7.0399×10^{-5}	2.00	2500
ROW3	1/100	1.5474×10^{-4}		1.7185×10^{-3}		690
	1/200	2.3165×10^{-5}	2.74	2.5534×10^{-4}	2.75	1036
	1/400	3.1047×10^{-6}	2.90	3.3251×10^{-5}	2.94	1712
	1/800	4.0081×10^{-7}	2.95	4.1768×10^{-6}	2.99	3004
ROW4	1/100	3.9102×10^{-6}		3.6770×10^{-5}		682
	1/200	2.5386×10^{-7}	3.95	2.3433×10^{-6}	3.97	1091
	1/400	2.5819×10^{-8}	3.30	1.3773×10^{-7}	4.09	1849
	1/800	7.5834×10^{-9}	1.78	1.7492×10^{-8}	2.98	3184
$tol_{rel,linear} = 10^{-8}$						
	$\Delta t/t_p$	$E_{L_2}(\rho)$	order	$E_{L_2}(u)$	order	Run time (seconds)
ROW2	1/100	Diverged	–	–	–	
	1/200	8.6824×10^{-5}		1.1229×10^{-3}		906
	1/400	2.1810×10^{-5}	1.99	2.8147×10^{-4}	2.00	1557
	1/800	5.4619×10^{-6}	2.00	7.0398×10^{-5}	2.00	2722
ROW3	1/100	1.5476×10^{-4}		1.7186×10^{-3}		862
	1/200	2.3158×10^{-5}	2.74	2.5535×10^{-4}	2.75	1589
	1/400	3.1015×10^{-6}	2.90	3.3242×10^{-5}	2.94	2149
	1/800	4.0093×10^{-7}	2.95	4.2024×10^{-6}	2.98	3481
ROW4	1/100	3.9055×10^{-6}		3.6728×10^{-5}		1186
	1/200	2.5299×10^{-7}	3.96	2.3382×10^{-6}	3.97	1381
	1/400	2.6107×10^{-8}	3.28	1.4352×10^{-7}	4.03	2237
	1/800	7.5988×10^{-9}	1.78	1.7442×10^{-8}	3.04	3720

End of Table 3.

4.1.3 Comparison of different time integrators

The order of accuracy study of ESDIRK and BDF2 and linearly implicit ROW are summarized in Figure 1. For ESDIRK and BDF2, $tol^{ESDIRK,BDF2}_{rel,nonlinear} = 10^{-6}$ and $tol^{ESDIRK,BDF2}_{rel,linear} = 10^{-1}$. $tol^{ROW}_{rel,linear} = 10^{-6}$ is employed for ROW.

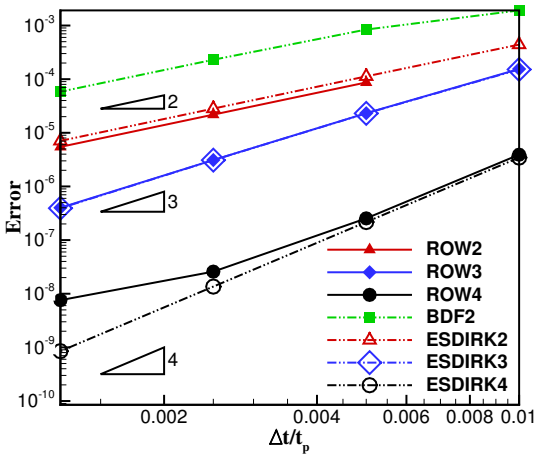
Figure 1(b) presents the run time versus errors of different time integrators with different convergence criteria. As illustrated in Figure 1(b), all multistage methods are significantly more efficient than BDF2. We notice that ROW2 and ESDIRK2 intersect with ROW3 and ESDIRK3. However, as the error threshold is decreased, higher-order methods will be more efficient.

When $tol^{ESDIRK}_{rel,nonlinear}$ is the same as $tol^{ROW}_{rel,linear}$, ROW methods are more efficient than ESDIRK methods. However, ROW methods cannot preserve the nominal order of accuracy when the convergence criterion is not tight, and even suffer from severe order reduction [40]. Instead, $tol_{rel,nonlinear} = 10^{-4}$ can make ESDIRK methods preserve the nominal order of accuracy (0.46 order reduction at most). It is observed that when $tol^{ROW}_{rel,linear} = 10^{-6}$ and $tol^{ESDIRK}_{rel,nonlinear} = 10^{-4}$ are employed, the ESDIRK method is more efficient than the ROW method when they have the same order of accuracy and number of stages. This indicates that the ESDIRK method tends to be over-solved more easily than the ROW method if the nonlinear convergence criterion $tol^{ESDIRK}_{rel,nonlinear}$ is not set up judiciously. We also note that ESDIRK can be more robust than ROW. As documented in Table 1 and Table 3, when the tolerance criteria are set to 10^{-2}, ESDIRK2 can show optimal convergence rate, while ROW cannot; when $\Delta t = t_p/100$, ROW2 even fails to converge.

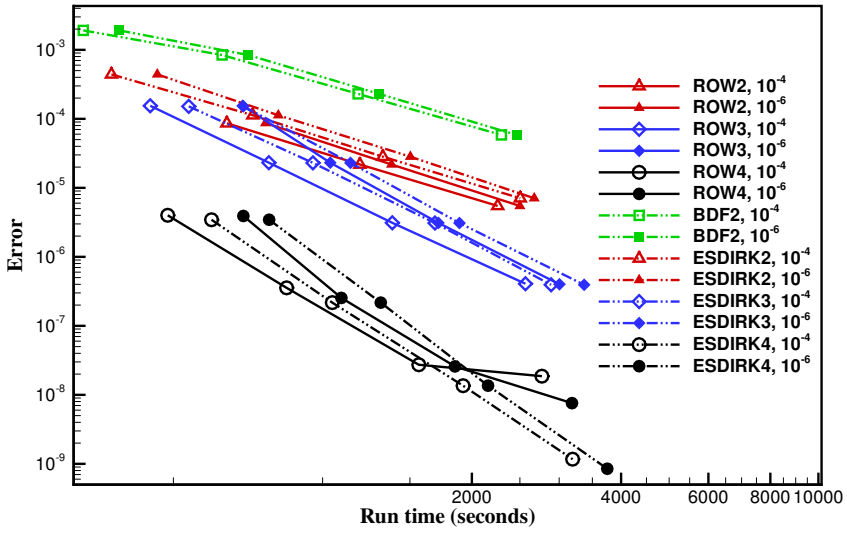
4.2 Laminar flow over a circular cylinder

In this subsection, we employ laminar flow over the circular cylinder as an example to study the performance of different time integrators. This case has been tested in various literature [21, 45]. The Reynolds number of the inflow with respect to the diameter of the cylinder is $\mathrm{Re}_d = 1200$, and the Mach number is $\mathrm{Ma} = 0.1$. The diameter of the cylinder is $d = 1$, and the computational domain is $[-100, 200] \times [-100, 100]$. The mesh in the near wall region and the wake region, and an instance of the vortex shedding are presented in Figure 2. There are 5690 elements in the mesh. The height of the first layer of the mesh is roughly 0.0033. The P^3 FR method is employed for spatial discretization. 16 processes are used for this case. $\Delta\tau_{init} = 0.01$ and $\Delta\tau_{max} = 1.0$ are used for all numerical experiments in this section. $tol^{ROW}_{rel,linear} = tol^{ESDIRK,BDF2}_{rel,nonlinear} = 10^{-6}$, and $tol^{ESDIRK,BDF2}_{rel,linear} = 10^{-1}$. The maximum number of iterations for GMRES is 500. As aforementioned, for ESDIRK and BDF2, if the linear solver fails to drive the residual to drop by one order of magnitude, the current iteration in the pseudo-transient continuation procedure will decrease $\Delta\tau_{max}$ by half and restart.

The flow is initialized with the steady solution when $\mathrm{Re} = 40$. Then, we use ESDIRK4 to run this simulation untill $t = 180$ with $\Delta t = 0.001$ to obtain the initial conditions for the convergence and efficiency study. For the convergence and efficiency study, we run all simulations for ten convective time units. The time step size Δt is refined from 0.2 to 0.00625. We use the numerical results of explicit SSPRK3 with a small time step $\Delta t = 5 \times 10^{-6}$ as the reference. The drag coefficient C_d is used for the error estimation.



(a)



(b)

Figure 1: (a) The convergence study of different time integrators and (b) efficiency study of different time integrators for the vortex propagation.

The error is calculated as

$$Error(C_d) = \sqrt{\frac{\sum_{n=1}^{N}(C_{d,n} - C_{d,ref,n})^2}{N}}, \tag{35}$$

where $C_{d,ref,n}$ is the reference value from SSPRK3, and N is the number of time steps. The results from convergence and efficiency study are presented in Figure 3(a) and Figure 3(b), respectively. As the time step size Δt is refined, all second- and third-order methods will converge at the nominal convergence rate. For both ROW4 and ESDIRK4, we have observed order reduction. The order reduction of ROW4 is more severe than that of ESDIRK4. In terms of run time, when $\Delta t = 0.2$, unexpected computational cost is observed for ESDIRK and BDF2. Many iterations in the pseudo-transient continuation are rejected due to the poor performance of the element-Jacobi preconditioner. However, all ROW methods fail when $\Delta t = 0.2$ and ROW2 even fails when $\Delta t = 0.1$. At a relatively large error level, such as 10^{-3}, second-order methods take the least amount of time. However, to reach a lower error level, higher-order methods are more efficient. We have noticed that the abnormal increase in run time for ROW4 when $\Delta t = 0.05$ and $\Delta t = 0.025$. This is due to the fact that the residual of the restarted GMRES solver with an element-Jacobi preconditioner sometimes cannot converge to the designated $tol^{ROW}_{rel,linear}$ when the maximum number of iteration is reached. From this study, we find that the performance of schemes from the ESDIRK family is more consistent than that of ROW methods when simulating unsteady flows over walls. Due to the limitation of the element-Jacobi preconditioner, ROW is only found to be consistently more efficient when the time step size Δt is small enough such that the GMRES solver can converge to the designated $tol^{ROW}_{rel,linear}$ within the maximum number of iterations.

4.3 Taylor–Green vortex

The Taylor–Green vortex is a benchmark to test the accuracy and performance of high-order methods on the direct numerical simulation of a 3D periodic and transitional flow defined by initial conditions [42]

$$\begin{cases} u = \ V_0 \sin(x/L)\cos(y/L)\cos(z/L), \\ v = -V_0 \cos(x/L)\sin(y/L)\cos(z/L), \\ w = 0, \\ p = p_0 + \frac{\rho_0 V_0^2}{16}\left(\cos(2x/L) + \cos(2y/L)\right)\left(\cos(2z/L) + 2\right). \end{cases} \tag{36}$$

The domain is $\Omega = [-\pi L, \pi L]^3$. The Reynolds number of the flow is defined as $\mathrm{Re} = \frac{\rho_0 V_0 L}{\mu}$ and is equal to 1600. For this study, we consider the flow with weak compressibility and the perfect gas law holds, i.e., $p = \rho RT$. The

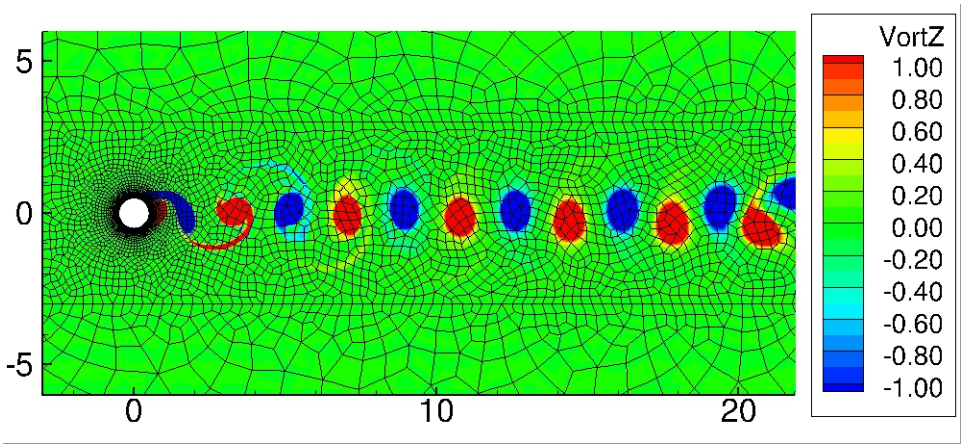
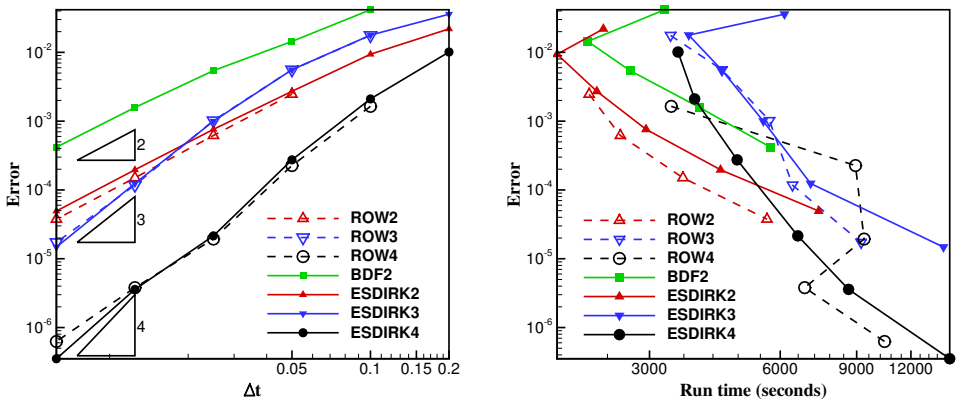


Figure 2: The mesh and an instance of the wake of vortex shedding for the laminar flow over a circular cylinder when $\mathrm{Re}_d = 1200$.



(a) (b)

Figure 3: The convergence and efficiency study for the laminar flow over the circular cylinder. (a) Error vs. the time step size Δt and (b) error vs. run time.

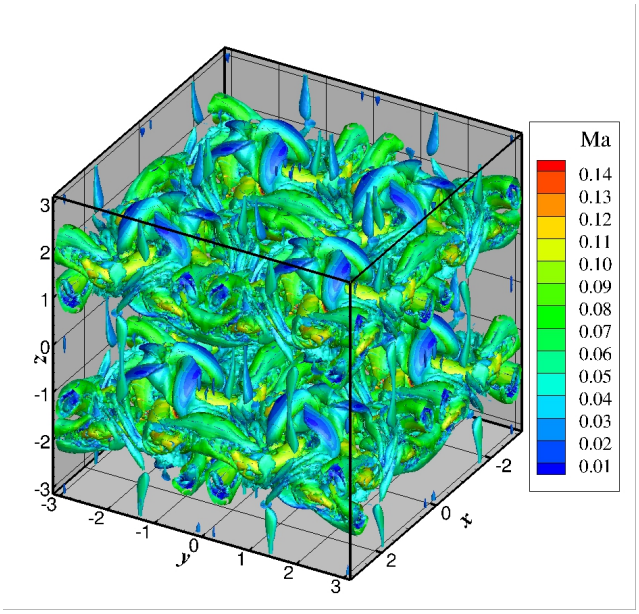


Figure 4: The isosurface of the Q criterion, where $Q_{criterion} = 1$, colored by Ma at $t = 8t_c$ for Taylor–Green vortex evolution.

Prandtl number is $\mathrm{Pr} = \frac{\mu c_p}{\kappa} = 0.71$. We assume that the gas has zero bulk viscosity $\mu_v = 0$. The Mach number $\mathrm{Ma} = \frac{V_0}{c_0} = 0.1$, where c_0 is the speed of sound corresponding to p_0. The characteristic convection time is defined as $t_c = \frac{L}{V_0}$. The maximum dissipation occurs at $t \approx 8t_c$. An uniform 64^3 mesh is employed and the P^3 FR methods is used for the spatial discretization. 512 processes are employed to conduct the numerical experiments. The numerical simulation is conducted until $t = 10t_c$. Figure 4 presents the isosurface of the Q criterion, where $Q_{criterion} = 1$, colored by Ma at $t = 8t_c$.

We employ the error of kinetic energy dissipation rate for the accuracy and efficiency study. The kinetic energy dissipation rate of compressible flows is the summation of three parts as $\epsilon = \epsilon_1 + \epsilon_2 + \epsilon_3$:

$$\epsilon_1 = 2\frac{\mu}{\rho_0}\frac{1}{V}\int_\Omega \boldsymbol{S}^d : \boldsymbol{S}^d V, \tag{37}$$

where $\boldsymbol{S}^d$ is the deviatoric part of the strain rate tensor, and V is the volume of the domain Ω,

$$\epsilon_2 = \frac{\mu_v}{\rho_0}\frac{1}{V}\int_\Omega (\nabla \cdot \boldsymbol{v})^2 dV, \tag{38}$$

where $\mu_v = 0$, and

$$\epsilon_3 = -\frac{1}{\rho_0}\frac{1}{V}\int_\Omega p\nabla \cdot \boldsymbol{v} dV. \tag{39}$$

The numerical results of SSPRK3 method with $\Delta t = 2 \times 10^{-4} t_c$ is adopted as the reference data for error evaluations. We define the error of the kinetic

energy dissipation rate as

$$Error(\epsilon) = \sqrt{\frac{\sum_{n=1}^{N}(\epsilon_n - \epsilon_{ref,n})^2}{N}}, \tag{40}$$

where $\epsilon_{ref,n}$ is the reference value from SSPRK3, and N is the number of time steps.

As discovered in previous sections, the convergence criteria have significant effect on the efficiency of ROW, ESDIRK and BDF2. In this section, we only consider $tol_{rel,linear}$ of ROW is the same as $tol_{rel,nolinear}$ of ESDIRK and BDF2. Herein, $tol^{ROW}_{rel,linear} = tol^{ESDIRK,BDF2}_{rel,nonlinear} = 10^{-6}$. For the inexactly linear-solving part of ESDIRK and BDF2, we employ $tol^{ESDIRK,BDF2}_{rel,linear} = 10^{-1}$ to save computational cost. The maximum number of iterations for GMRES is 600, which is large enough to guarantee that GMRES can converge to the designated tolerance for all implicit time integrators.

The time step size Δt is refined from $t_c/25$ to $t_c/100$. The kinetic energy dissipation rate history when $\Delta t = t_c/25$ is presented in Figure 5(a). A close-up view within $t/t_c \in [8.5, 10]$ is illustrated in Figure 5(b). The numerical results from the spectral method on a 512^3 mesh is also presented for reference [56]. Our observation is that the results of ROW4 and ESDIRK4 almost coincide with that of SSPRK3. BDF2 is much more dissipative than ROW2 and ESDIRK2. The convergence study is presented in Figure 6. Figure 6(a) shows the error vs. time step size $\Delta t/t_c$ and Figure 6(b) shows the error vs. run time. From Figure 6(a), the convergence features of ROW and ESDIRK are almost the same. As is shown in Figure 6(b), BDF2 is not as efficient as ROW2 and ESDIRK2 as expected. When $tol^{ROW}_{rel,linear} = tol^{ESDIRK,BDF2}_{rel,nonlinear} = 10^{-6}$, ROW methods are found to be more efficient than ESDIRK methods. From Figure 5, we find that for turbulent simulation, compared to the results from SSPRK3 with very small time steps, all time integrators with excessively large time step size will lead to numerical dissipation of the kinetic energy dissipation rate except ROW4 and ESDIRK4. This reveals that the dissipation due to the temporal discretization should also be taken into account for turbulent simulation. We refers interested readers to Refs. [18, 19] for more discussions.

4.4 Transitional flow over the SD7003 wing

The transitional flow over the SD7003 wing is studied when the Reynolds number of the inflow with respect to the chord length of the wing is $\mathrm{Re}_c = 60000$ and the angle of attack (AoA) of the inflow is 8°. The Mach number is $\mathrm{Ma} = 0.1$. The geometry is obtained from the 1st International Workshop on High-Order CFD Methods [57]. The chord length is $c = 1$ with the sharp trailing edge rounded by an arc of radius $r \approx 0.0004c$. The unstructured

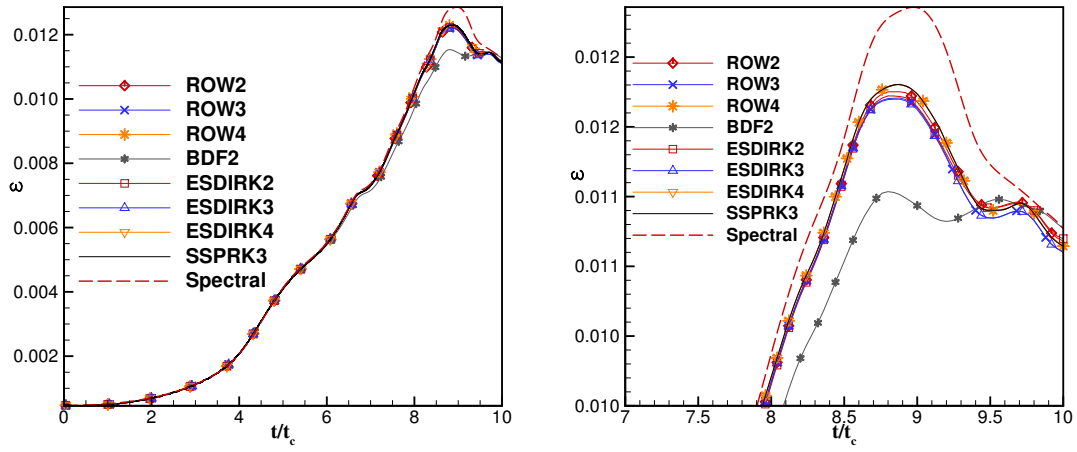


Figure 5: The kinetic energy dissipation rate history of Taylor–Green vortex decaying. (a) A full view when $t/t_c \in [0, 10]$; (b) a close-up view when $t/t_c \in [7, 10]$.

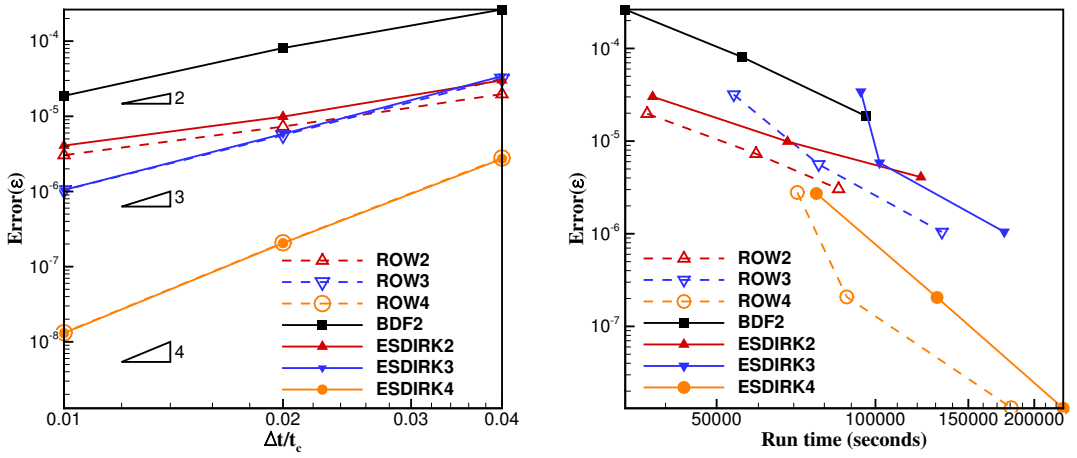


Figure 6: The convergence and efficiency study for Taylor–Green vortex evolution. (a) Error vs. time step size $\Delta t/t_c$ and (b) error vs. run time.

mesh near the SD7003 wing is illustrated in Figure 7. The height of the first layer close to the wall is $0.0003c$. The 3D mesh is obtained by extruding a 2D mesh along the z-direction for 20 layers and each layer has a thickness of $0.01c$. There are 109540 quadric hexahedral elements. The third-order FR method is employed for spatial discretization. 252 processes are used for the parallel simulation in this section.

We compare the computational cost of BDF2, ESDIRK2, ROW2 and the explicit SSPRK3. We run all simulations until $t_{end} = 32$ and the instantaneous solutions in $t \in (26, 32]$ are averaged for statistics. For BDF2 and ESDIRK2, $\Delta t = 0.002$ is used and the flow field is initialized uniformly with the inflow conditions. ROW2 fails when the time step Δt equals to 0.002. Therefore, we use a smaller time step size $\Delta t = 0.001$ for ROW2. In this section, $tol^{ROW}_{rel,linear} = tol^{ESDIRK,BDF2}_{rel,nonlinear} = 10^{-4}$ and $tol^{ESDIRK,BDF2}_{rel,linear} = 10^{-1}$. $\Delta\tau_{init} = 0.0002$ and $\Delta\tau_{max} = 0.004$ are used for ESDIRK2 and BDF2. The maximum number of iterations for the GMRES solver is 200. We use SSPRK3 to run this simulation for 32 convective time units with $\Delta t = 2 \times 10^{-6}$, which is slightly smaller than the allowed maximum time step size, as a reference. Note that the time step for the explicit SSPRK3 method is about three orders of magnitude smaller than that for the implicit methods due to the highly anisotropic mesh near the wing.

The instantaneous isosurface of Q criterion, where $Q_{criterion} = 500$, and the averaged field of the velocity component in the x-direction are plotted in Figure 8(a) and Figure 8(b), respectively. Numerical predictions, namely, the time-averaged lift coefficient C_l, drag coefficient C_d, the separation point x_s, and the reattachment point x_{re}, are documented in Table 4. The predictions of different time integration methods are close to each other. Compared to previous numerical and experimental results, a decent agreement has been observed. Run time of all methods is also provided. Due to the difficulty of determining the accuracy of different time integration methods via inspecting the averaged values, we examine the run time only. It is observed that all implicit time integration methods are significantly faster than the explicit method. Up to 82.36% computational cost can be saved by employing an implicit time integrator instead of using an explicit one. Not surprisingly, BDF2 takes the largest amount of run time to finish the simulation. Even though ROW2 diverged when $\Delta t = 0.002$, it can still outperform BDF2 with a smaller $\Delta t = 0.001$ and is only slightly more expensive than ESDIRK2. Note that the total number of time steps of ROW is twice as that of ESDIRK2 and BDF2. This indicates that when the time step size is refined to a suitable level, i.e., the linear solver can easily converge to preserve the accuracy of ROW, the performance of ROW is comparable with that of ESDIRK.

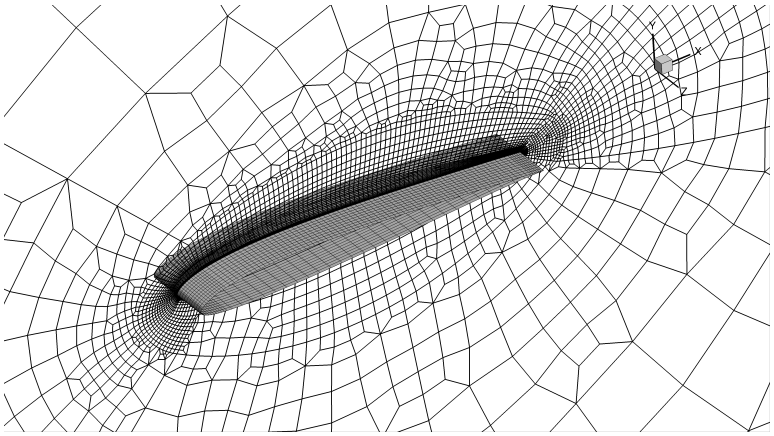
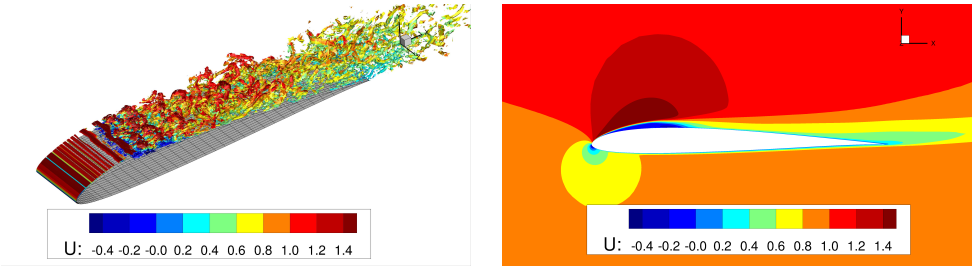


Figure 7: Unstructured meshes near the SD7003 wing.



(a) (b)

Figure 8: (a) The isosurface of Q criterion, where $Q_{criterion} = 500$, at $t = 32$ and (b) averaged velocity component in the x-direction using ESDIRK2.

Table 4: A summary of flow statistics from long-time simulation of the transitional flow over SD7003. The abbreviation Inc. stands for incompressible and Exp. stands for experiment.

Time Marching	Ma	C_l	C_d	x_s	x_{re}	Run time (hours)
BDF2 (current work)	0.1	0.9212	0.0476	0.0301	0.3290	58.77
ESDIRK2 (current work)	0.1	0.9191	0.0474	0.0302	0.3258	56.06
ROW2 (current work)	0.1	0.9236	0.0459	0.0293	0.3119	57.02
SSPRK3 (current work)	0.1	0.9201	0.0463	0.0306	0.3216	317.75
Beck et al. [58]	0.1	0.923	0.045	0.027	0.310	
Galbraith &Visbal [59]	0.1	0.91	0.043	0.04	0.28	
Bassi et al. [41]	Inc.	0.953	0.045	0.027	0.294	
Selig et al. [60]	Exp.	0.92	0.029			

5 Conclusions

In this work, we compare the accuracy and efficiency of ROW and ESDIRK (from second order to fourth order) and BDF2 for unsteady flow simulation with the high-order FR formulation. We find that ROW2 and ESDIRK2 are more computationally efficient than BDF2. To achieve an accurate estimation, i.e., the target error is small, higher-order implicit time integrators are more efficient than lower-order ones.

The efficiency of ROW and ESDIRK highly depends on the convergence criteria of solving nonlinear and linear equations. For most of the case, when $tol^{ROW}_{rel,linear} = tol^{ESDIRK}_{rel,nonlinear}$ and the time step size is refined, ROW is more efficient since it only needs to solve one linear system at each stage. However, the tolerance $tol^{ROW}_{rel,linear}$ needs to be sufficiently small to preserve the order of accuracy of ROW. With a large time step size, to drive the residual of GMRES to a trivial value is expensive with the element-Jacobi preconditioner for wall-bounded flows; this may degrade the advantage of ROW over ESDIRK. Besides, the ROW method is more prone to suffer from order reduction. On the contrary, the tolerance $tol^{ESDIRK}_{rel,nonlinear}$ for the pseudo-time iterations in ESDIRK can be relatively larger than $tol^{ROW}_{rel,linear}$ to preserve the nominal order of accuracy. Therefore, when $tol^{ESDIRK}_{rel,nonlinear}$ is allowed to be larger than $tol^{ROW}_{rel,linear}$, ESDIRK can be more efficient. The inexact Newton's method, i.e., the pseudo-transient continuation, gives ESDIRK the edge that the stiffness of the linearized system can be controlled with the pseudo time step size. A relaxed GMRES tolerance, such as $tol^{ESDIRK}_{rel,linear} = 10^{-1}$, can be used to accelerate simulation. This feature makes ESDIRK more robust and allow for a larger time step size than ROW.

We note that the preconditioner has a significant impact on the performance of implicit time integrators. In this work, only the element-Jacobi preconditioner is considered. Future work will be to develop low-storage preconditioners, such as the p-multigrid preconditioner, to improve the efficiency of implicit time marching methods.

Acknowledgment

The authors gratefully acknowledge the support of the Office of Naval Research through the award N00014-16-1-2735, and the faculty startup support from the department of mechanical engineering at the University of Maryland, Baltimore County (UMBC). The hardware used in the computational studies is part of the UMBC High Performance Computing Facility (HPCF). The facility is supported by the U.S. National Science Foundation through the MRI program (grant nos. CNS-0821258, CNS-1228778, and OAC-1726023) and the SCREMS program (grant no. DMS-0821311), with additional substantial support from UMBC.

Appendix

The coefficients of ESDIRK and ROW methods investigated in this study are documented here for completeness.

Table 5: Coefficients of ESDIRK methods

ESDIRK2 [33]		
ω	=	0.2928932188134524
a_{21}	=	0.2928932188134524
a_{31}	=	0.3535533905932738
a_{32}	=	0.3535533905932738
b_1	=	0.3535533905932738
b_2	=	0.3535533905932738
b_3	=	0.2928932188134524

ESDIRK3 [21]		
ω	=	0.4358665215084590
a_{21}	=	0.4358665215084590
a_{31}	=	0.2576482460664272
a_{32}	=	−0.0935147675748862
a_{41}	=	0.1876410243467238
a_{42}	=	−0.5952974735769549
a_{43}	=	0.9717899277217721
b_1	=	0.1876410243467238
b_2	=	−0.5952974735769549
b_3	=	0.9717899277217721
b_4	=	0.4358665215084590

ESDIRK4 [21]		
ω	=	0.25
a_{21}	=	0.25
a_{31}	=	0.137776
a_{32}	=	−0.055776
a_{41}	=	0.1446368660269822
a_{42}	=	−0.2239319076133447
a_{43}	=	0.4492950415863626
a_{51}	=	0.0982587832835648
a_{52}	=	−0.5915442428196704
a_{53}	=	0.8101210538282996
a_{54}	=	0.2831644057078060
a_{61}	=	0.1579162951616714
a_{62}	=	0.0
a_{63}	=	0.1867589405240008
a_{64}	=	0.6805652953093346
a_{65}	=	−0.2752405309950067
b_1	=	0.1579162951616714
b_2	=	0.0
b_3	=	0.1867589405240008
b_4	=	0.6805652953093346
b_5	=	−0.2752405309950067
b_6	=	0.25

Table 6: Coefficients of ROW methods

ROW2 [37]		
ω	=	0.2281554936539618
a_{21}	=	4.3829757679062376
a_{31}	=	4.3829757679062376
a_{32}	=	4.3829757679062376
c_{21}	=	−4.3829757679062376
c_{31}	=	−4.3829757679062376
c_{32}	=	−16.827500814147036
m_1	=	4.3829757679062377
m_2	=	4.3829757679062377
m_3	=	1.0

ROW3 [36]		
ω	=	0.4358665215084590
a_{21}	=	2.0
a_{31}	=	1.4192173174557646
a_{32}	=	−0.2592322116729697
a_{41}	=	4.1847604823191607
a_{42}	=	−0.2851920173554959
a_{43}	=	2.2942803602790417
c_{21}	=	−4.5885607205580834
c_{31}	=	−4.1847604823191607
c_{32}	=	0.2851920173554959
c_{41}	=	−6.3681792001283574
c_{42}	=	−6.7956209444668360
c_{43}	=	2.8700986043310560
m_1	=	4.1847604823191602
m_2	=	−0.2851920173554959
m_3	=	2.2942803602790414
m_4	=	1.0

ROW4 [38]		
ω	=	0.25
a_{21}	=	3.0
a_{31}	=	1.831036793486759
a_{32}	=	0.495518396743379
a_{41}	=	2.304376582692669
a_{42}	=	−0.052492752457430
a_{43}	=	−1.176798761832782
a_{51}	=	−7.170454962423024
a_{52}	=	−4.741636671481785
a_{53}	=	−16.31002631330971
a_{54}	=	−1.062004044111401
a_{61}	=	−7.170454962423024
a_{62}	=	−4.741636671481785
a_{63}	=	−16.31002631330971
a_{64}	=	−1.062004044111401
a_{65}	=	1.0
c_{21}	=	3.0
c_{31}	=	1.831036793486759
c_{32}	=	0.495518396743379
c_{41}	=	2.304376582692669
c_{42}	=	−0.052492752457430
c_{43}	=	−1.176798761832782
c_{51}	=	−7.170454962423024
c_{52}	=	−4.741636671481785
c_{53}	=	−16.31002631330971
c_{54}	=	−1.062004044111401
c_{61}	=	−7.170454962423024
c_{62}	=	−4.741636671481785
c_{63}	=	−16.31002631330971
c_{64}	=	−1.062004044111401
c_{65}	=	1.0
m_1	=	−7.170454962423024
m_2	=	−4.741636671481785
m_3	=	−16.31002631330971
m_4	=	−1.062004044111401
m_5	=	1.0
m_6	=	1.0

References

[1] B. Cockburn and C.-W. Shu, "TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework," *Math. Comput.*, vol. 52, pp. 411–435, 1989.

[2] F. Bassi and S. Rebay, "A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations," *Journal of computational physics*, vol. 131, no. 2, pp. 267–279, 1997.

[3] J. S. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. New York: Springer-Verlag, 2008.

[4] H. T. Huynh, "A flux reconstruction approach to high-order schemes including discontinuous Galerkin methods," in *the 18th AIAA Computational Fluid Dynamics Conference*, (Miami, FL), 2007. AIAA-2007-4079.

[5] H. T. Huynh, "A reconstruction approach to high-order schemes including discontinuous Galerkin methods for diffusion," in *the 47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace*, (Orlando, FL), 2009. AIAA-2009-403.

[6] Z. J. Wang and H. Y. Gao, "A unifying lifting collocation penalty formulation including the discontinuous Galerkin, spectral volume/difference methods for conservation laws on mixed grids," *Journal of Computational Physics*, vol. 228, pp. 8161–8186, 2009.

[7] P. E. Vincent, P. Castonguay and A. Jameson, "A new class of high-order energy stable flux reconstruction schemes," *Journal of Scientific Computing*, vol. 47, pp. 50–72, 2011.

[8] F. Bassi, A. Crivellini, Stefano Rebay, and M. Savini, "Discontinuous Galerkin solution of the Reynolds-averaged Navier–Stokes and k–ω turbulence model equations," *Computer & Fluids*, vol. 34, pp. 507–540, 2005.

[9] C. Liang, S. Premasuthana, A. Jameson, and Z. J. Wang, "Large Eddy Simulation of Compressible Turbulent Channel Flow with Spectral Difference method," in *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, 2009. AIAA-2009-402.

[10] A. Uranga and P.-O. Persson and M. Drela and J. Peraire, "Implicit Large Eddy Simulation of transition to turbulence at low Reynolds

numbers using a Discontinuous Galerkin method," *International Journal for Numerical Methods in Engineering*, vol. 87, pp. 232—-261, 2011.

[11] F. Bassi, L. Botti, A. Colombo, A. Crivellini, A. Ghidoni, and F. Massa, "On the development of an implicit high-order Discontinuous Galerkin method for DNS and implicit LES of turbulent flows," *European Journal of Mechanics-B/Fluids*, vol. 55, pp. 367–379, 2016.

[12] M. A. Ceze and K. J. Fidkowski, "High-Order Output-Based Adaptive Simulations of Turbulent Flow in Two Dimensions," *AIAA Journal*, vol. 54, pp. 2611–2625, 2016.

[13] Z.J. Wang and Y. Li and F. Jia and G. M. Laskowski and J. Kopriva and U. Paliath and R. Bhaskaran, "Towards industrial large eddy simulation using the FR/CPR method," *Computers & Fluids*, vol. 156, pp. 579–589, 2017.

[14] J. S. Park and F. D. Witherden and P. E. Vincent, "High-Order Implicit Large-Eddy Simulations of Flow over a NACA0021 Aerofoil," *AIAA Journal*, vol. 55, pp. 2186–2197, 2017.

[15] B. R. Ahrabi, M. J. Brazell, and D. J. Mavriplis, "An Investigation of Continuous and Discontinuous Finite-Element Discretizations on Benchmark 3D Turbulent Flows," in *2018 AIAA Aerospace Sciences Meeting*, 2018. AIAA-2018-1569.

[16] S. Gottlieb, C.-W. Shu, and E. Tadmor, "Strong stability-preserving high-order time discretization methods," *SIAM review*, vol. 43, no. 1, pp. 89–112, 2001.

[17] H. Yang, F. Li, and J. Qiu, "Dispersion and Dissipation Errors of Two Fully Discrete Discontinuous Galerkin Methods," *Journal of Scientific Computing*, vol. 55, pp. 552–574, 2013.

[18] B. Vermeire and P. Vincent, "On the behaviour of fully-discrete flux reconstruction schemes," *Computer Methods in Applied Mechanics and Engineering*, vol. 315, pp. 1053–1079, 2017.

[19] M. Alhawwary and Z. Wang, "Fourier analysis and evaluation of DG, FD and compact difference methods for conservation laws," *Journal of Computational Physics*, vol. 373, pp. 835–862, 2018.

[20] G. Wanner and E. Hairer, *Solving ordinary differential equations II*. Springer Berlin Heidelberg, 1996.

[21] H. Bijl, M. H. Carpenter, V. N. Vatsa, and C. A. Kennedy, "Implicit time integration schemes for the unsteady compressible Navier–Stokes equations: laminar flow," *Journal of Computational Physics*, vol. 179, no. 1, pp. 313–329, 2002.

[22] M. H. Carpenter, S. A. Viken, and E. J. Nielsen, "The efficiency of high order temporal schemes," *AIAA Paper*, vol. 86, 2003.

[23] L. Wang and D. J. Mavriplis, "Implicit solution of the unsteady Euler equations for high-order accurate discontinuous Galerkin discretizations," *Journal of Computational Physics*, vol. 225, no. 2, pp. 1994–2015, 2007.

[24] J. Cash, "On the integration of stiff systems of ODEs using extended backward differentiation formulae," *Numerische Mathematik*, vol. 34, no. 3, pp. 235–246, 1980.

[25] D. A. Voss and M. J. Casper, "Efficient split linear multistep methods for stiff ordinary differential equations," *SIAM Journal on Scientific and Statistical Computing*, vol. 10, no. 5, pp. 990–999, 1989.

[26] G.-Y. Psihoyios and J. Cash, "A stability result for general linear methods with characteristic function having real poles only," *BIT Numerical Mathematics*, vol. 38, no. 3, pp. 612–617, 1998.

[27] G. Psihoyios, "A general formula for the stability functions of a group of implicit advanced step-point (IAS) methods," *Mathematical and computer modelling*, vol. 46, no. 1-2, pp. 214–224, 2007.

[28] A. Nigro, A. Ghidoni, S. Rebay, and F. Bassi, "Modified extended BDF scheme for the discontinuous Galerkin solution of unsteady compressible flows," *International Journal for Numerical Methods in Fluids*, vol. 76, no. 9, pp. 549–574, 2014.

[29] A. Nigro, C. De Bartolo, F. Bassi, and A. Ghidoni, "Up to sixth-order accurate A-stable implicit schemes applied to the discontinuous Galerkin discretized Navier–Stokes equations," *Journal of Computational Physics*, vol. 276, pp. 136–162, 2014.

[30] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*. Chichester: Wiley, 2002.

[31] A. Jameson, "Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings," in *10th Computational Fluid Dynamics Conference*, p. 1596, 1991.

[32] A. Jameson, "Evaluation of Fully Implicit Runge Kutta Schemes for Unsteady Flow Calculations," *Journal of Scientific Computing*, vol. 73, no. 2-3, pp. 819–852, 2017.

[33] C. A. Kennedy and M. H. Carpenter, "Diagonally Implicit Runge-Kutta Methods for Ordinary Differential Equations. A Review," Tech. Rep. NASA/TM–2016–219173, NASA, 2016.

[34] A. J. Baker and G. S. Iannelli, "A stiffly-stable implicit Runge-Kutta algorithm for CFD applications," in *26th AIAA Aerospace Sciences Meeting*, 1988. AIAA Paper 88-0416.

[35] J. Lang and J. Verwer, "ROS3P–an accurate third-order Rosenbrock solver designed for parabolic problems," *BIT Numerical Mathematics*, vol. 41, no. 4, pp. 731–738, 2001.

[36] J. Rang and L. Angermann, "New Rosenbrock W-methods of order 3 for partial differential algebraic equations of index 1," *BIT Numerical Mathematics*, vol. 45, no. 4, pp. 761–787, 2005.

[37] J. Rang, "An analysis of the Prothero–Robinson example for constructing new DIRK and ROW methods," *Journal of Computational and Applied Mathematics*, vol. 262, pp. 105–114, 2014.

[38] G. Steinebach, "Order-reduction of ROW-methods for DAEs and method of lines applications," *Preprint/Fachbereich Mathematik, Technische Hochschule Darmstadt*, vol. 1741, 1995.

[39] P. Tranquilli and A. Sandu, "Rosenbrock–Krylov Methods for Large Systems of Differential Equations," *SIAM Journal on Scientific Computing*, vol. 36, no. 3, pp. A1313–A1338, 2014.

[40] J. Rang, "Improved traditional Rosenbrock–Wanner methods for stiff ODEs and DAEs," *Journal of Computational and Applied Mathematics*, vol. 286, pp. 128–144, 2015.

[41] F. Bassi, L. Botti, A. Colombo, A Ghidoni and F. Massa, "Linearly implicit Rosenbrock-type Runge–Kutta schemes applied to the Discontinuous Galerkin solution of compressible and incompressible unsteady flows," *Computers & Fluids*, vol. 118, pp. 305–320, 2015.

[42] X. Liu, Y. Xia, H. Luo, and L. Xuan, "A comparative study of Rosenbrock-type and implicit Runge-Kutta time integration for discontinuous Galerkin method for unsteady 3D compressible Navier–Stokes equations," *Communications in Computational Physics*, vol. 20, no. 4, pp. 1016–1044, 2016.

[43] M. Franciolini, A. Crivellini, and A. Nigro, "On the efficiency of a matrix-free linearly implicit time integration strategy for high-order Discontinuous Galerkin solutions of incompressible turbulent flows," *Computers & Fluids*, vol. 159, pp. 276–294, 2017.

[44] L. Wang and M. Yu, "On the parallel implementation and performance study of high-order Rosenbrock-type implicit Runge-Kutta methods for the FR/CPR solutions of the Navier–Stokes equations," in *2018 AIAA Aerospace Sciences Meeting*, 2018. AIAA-2018-1095.

[45] D. S. Blom, P. Birken, H. Bijl, F. Kessels, A. Meister, and A. H. van Zuijlen, "A comparison of Rosenbrock and ESDIRK methods combined with iterative solvers for unsteady compressible flows," *Advances in Computational Mathematics*, vol. 42, no. 6, pp. 1401–1426, 2016.

[46] A. Sarshar, P. Tranquilli, B. Pickering, A. McCall, C. J. Roy, and A. Sandu, "A numerical investigation of matrix-free implicit time-stepping methods for large CFD simulations," *Computers & Fluids*, vol. 159, pp. 53–63, 2017.

[47] Persson, P.-O. and J. Peraire, "Newton-GMRES Preconditioning for Discontinuous Galerkin Discretizations of the Navier–Stokes Equations," *SIAM Journal on Scientific Computing*, vol. 30, pp. 2709–2733, 2008.

[48] P. L. Roe, "Approximate Riemann solvers, parameter vectors and difference schemes," *Journal of Computational Physics*, vol. 43, pp. 357–372, 1981.

[49] H. Gao, Z. Wang, and H. Huynh, "Differential formulation of discontinuous Galerkin and related methods for the Navier–Stokes equations," *Communications in Computational Physics*, vol. 13, no. 4, pp. 1013–1044, 2013.

[50] W. A. Mulder and B. Van Leer, "Experiments with implicit upwind methods for the Euler equations," *Journal of Computational Physics*, vol. 59, no. 2, pp. 232–246, 1985.

[51] D. A. Knoll and D. E. Keyes, "Jacobian-free Newton–Krylov methods: a survey of approaches and applications," *Journal of Computational Physics*, vol. 193, no. 2, pp. 357–397, 2004.

[52] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, "PETSc users manual," Tech. Rep. ANL-95/11 - Revision 3.7, Argonne National Laboratory, 2016.

[53] D. Sharov, H. Luo, J. Baum, and R. Löhner, "Implementation of unstructured grid GMRES+LU-SGS method on shared-memory, cache-based parallel computers," in *38th Aerospace Sciences Meeting and Exhibit*, p. 927, 2000.

[54] M. Franciolini, L. Botti, A. Colombo, and A. Crivellini, "p-Multigrid matrix-free discontinuous Galerkin solution strategies for the under-resolved simulation of incompressible turbulent flows," *arXiv preprint arXiv:1809.00866*, 2018.

[55] L. Wang and M. Yu, "An Implicit High-Order Preconditioned Flux Reconstruction Method for Low-Mach-Number Flow Simulation with Dynamic Meshes," *International Journal for Numerical Methods in Fluids*, 2019.

[56] W. M. Van Rees, A. Leonard, D. Pullin, and P. Koumoutsakos, "A comparison of vortex and pseudo-spectral methods for the simulation of periodic vortical flows at high reynolds numbers," *Journal of Computational Physics*, vol. 230, no. 8, pp. 2794–2805, 2011.

[57] Z. J. Wang and K. Fidkowski and R. Abgrall and F. Bassi and D. Caraeni and A. Cary and H. Deconinck and R. Hartmann and K. Hillewaert and H. T. Huynh and N. Kroll and G. May and P.-O. Persson and B. van Leer and M. Visbal, "High-order CFD methods: current status and perspective," *International Journal for Numerical Methods in Fluids*, vol. 72, pp. 811–845, 2013.

[58] A. D. Beck, T. Bolemann, D. Flad, H. Frank, G. J. Gassner, F. Hindenlang, and C.-D. Munz, "High-order discontinuous Galerkin spectral element methods for transitional and turbulent flow simulations," *International Journal for Numerical Methods in Fluids*, vol. 76, no. 8, pp. 522–548, 2014.

[59] M. Galbraith and M. Visbal, "Implicit large eddy simulation of low Reynolds number flow past the SD7003 airfoil," in *46th AIAA Aerospace Sciences Meeting and Exhibit*, p. 225, 2008.

[60] M. S. Selig, *Summary of low speed airfoil data Vol. 1.* SoarTech Publications, 1995.