



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

Lai Wang*, Meilin Yu†

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

Abstract

The Rosenbrock-type implicit Runge-Kutta (ROIRK) methods only require one Jacobian matrix evaluation per time step rather than per stage as other types of implicit Runge-Kutta (IRK) methods need. This feature makes ROIRK attractive for numerical simulations using implicit methods. We present the parallel implementation of several matrix-based ROIRK methods with flux reconstruction/correction procedure reconstruction (FR/CPR) formulations for solving the 3D Navier-Stokes equations. In this study, METIS has been utilized to partition the mesh in the preprocessing. The complex-step derivative approximation is employed to evaluate the Jacobi matrix, accurate to machine zero. The GMRES solver in the PETSc library is used to iteratively solve the linear system. The ROIRK methods have demonstrated high order of accuracy in numerical simulations. The scalability study reveals that the matrix-based ROIRK methods have good parallel efficiency. With the block Jacobi preconditioner, it is observed that the linear systems resulting from ROIRK3-3 are stiffer than those from ROIRK2-2 and ROIRK4-6. This makes the scalability of ROIRK3-3 worse than ROIRK2-2 and ROIRK4-6 taking the number of stages into account.

Key Words

Rosenbrock-type implicit Runge-Kutta methods; Flux reconstruction/correction procedure via reconstruction; Parallel implementation; Scalability.

*Graduate student, AIAA Student Member, email: bx58858@umbc.edu

†Assistant professor, AIAA Senior Member, email: mlyu@umbc.edu

1 Introduction

Simulating fluid flows that have a large disparity in spatial and temporal scales requires that both the spatial discretization and time integration are of high accuracy and high resolution. The FR/CPR method developed by Huynh [1, 2] uses correction functions to reconstruct the local flux polynomials to achieve high order of accuracy. In the FR/CPR approach, the flexibility of different correction functions allows it to recover several high order methods, such as the discontinuous Galerkin (DG) method, the spectral difference/volume (SD/SV) methods, and also allows researchers to develop new methods under this framework. It was then extended by Wang and Gao in 2009 [3] to simplex elements. Vincent et al. developed a family of energy stable FR/CPR methods in 2011 [4]. A simplified FR approach, i.e. the direct flux reconstruction method (DFR), was then developed by Romero et al. [5] in 2016. Wang and Yu employed the compact finite approach to directly reconstruct the spatial derivatives within an element, and demonstrated the identity of this approach (named CDFR) with the nodal FR-DG and DFR methods [6]. Overall, the FR/CPR-family methods are accurate, robust, and easy to implement. And the compact nature makes the FR/CPR method well-suited for the high performance parallel computing.

Among IRK methods, the diagonally IRK methods are arguably the most widely used to solve practical stiff problems since they are relatively easy to implement. They are characterized by a lower triangular A-matrix with at least one nonzero diagonal entry and are sometimes referred to as semi-implicit or semi-explicit Runge-Kutta methods [7]. A comprehensive review about IRK methods can be found in Ref. [7]. Research has been conducted to apply IRK methods to solve the unsteady Navier-Stokes equations [8, 9, 10, 11] with high-order spatial discretization methods. Through comparison with the popular backward differentiation formulas (BDFs), especially BDF1 and BDF2, it is recommended that IRK methods are more preferable for stiff problems [8, 10]. Even though BDFs are efficient and easy to implement, they suffer from not being self-starting and not A-stable beyond second-order accuracy.

Linearly implicit Rosenbrock methods avoid solving nonlinear systems every stage in some IRK methods, such as singly diagonally implicit Runge-Kutta methods (SDIRK) [12], and solve only one linear system at each stage [13]. Different variations of Rosenbrock-type IRK methods can be found in Refs. [14, 15, 16, 17, 18]. For traditional Rosenbrock-type IRK methods, the accuracy of the methods depend on the availability of the exact Jacobi. Rosenbrock-Wanner methods have been developed to preserve the order of accuracy with an approximation of the exact Jacobi. For the Rosenbrock-Wanner method, a better approximation will ensure better stability. The Rosenbrock-Krylov method [18] is developed based on a Krylov space solution of the linear systems and has substantially fewer order conditions than Rosenbrock-Wanner methods. A practical obstacle of applying the ROIRK methods for solving large CFD problems is that the Jacobi matrix and the preconditioner matrix require a significant amount of memory. To overcome this trouble, the matrix-free approach can be employed. Conventionally, the finite difference approach is used for the matrix-vector approximation. This approximation

will degenerate the accuracy of traditional Rosenbrock-type methods. However, the Rosenbrock-Wanner method and Rosenbrock-Krylov method can avoid this problem. A more profound discussion on the matrix-free implementation of Rosenbrock-type IRK methods can be found in Ref. [19].

The applications of ROIRK with high-order spatial discretization methods have been popular in recent years. Bassi et al. have systematically investigated the performance of ROIRK methods up to sixth order on solving both compressible and incompressible Navier-Stokes equations with DG spatial discretizations [20, 21] and applied them for the DNS and ILES simulations. A comparative study of ROIRK and ESDIRK methods for the Navier-Stokes equations has been conducted by Liu et al. in the context of reconstructed Discontinuous Galerkin (rDG) methods [22].

In this work, we present the parallel implementation details for ROIRK methods up to fourth order with high-order FR/CPR methods on solving the 3D unsteady Navier-Stokes equations. Currently, the implementation is matrix-based under the circumstance that ‘exact’ Jacobi will be supplied. We focus on the scalability study of different ROIRK methods. An open-source software-package, METIS [23], is employed for mesh partitioning. The complex-step derivative approach [24] is adopted for the Jacobian matrix evaluation. This approach can provide more accurate(to machine zero) Jacobi evaluation when compared to the finite difference approach and is easier to implement or debug when compared to the analytical differentiation approach or the automatic differentiation approach. The GMRES solver in the PETSc library [25] serves as the iterative linear solver. The intention of this study is to provide a concise guide to researchers who are interested in applying FR/CPR methods and ROIRK methods with parallelism for solving the Navier-Stokes equations.

The remainder of this paper is organized as follows: Section 2 gives a brief introduction of the governing equations. The spatial discretization and time integration are presented in Section 3 and Section 4, respectively. Section 5 explains the parallel implementation in detail. Numerical results are demonstrated in Section 6. The work is briefly summarized in Section 7.

2 Governing equations

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

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{f} = 0, \quad (1)$$

where $\mathbf{q} = (\rho, \rho u_j, E)^T$, $j = 1, 2, 3$ are the conservative variables, $p = \rho RT$, $E = \frac{p}{\gamma-1} + \frac{1}{2}\rho(u_k u_k)$, R is the ideal gas constant, and γ is the specific heat ratio defined as $\gamma = C_p/C_v$. In this study, γ is set as 1.4. The flux \mathbf{f} consists of the inviscid part and viscous part, which can be expressed as $\mathbf{f} = \mathbf{f}_{inv}(\mathbf{q}) - \mathbf{f}_{vis}(\mathbf{q}, \nabla \mathbf{q})$. The component

forms of $\mathbf{f}_{inv}(\mathbf{q})$ and $\mathbf{f}_{vis}(\mathbf{q}, \nabla \mathbf{q})$ are formulated as

$$\mathbf{f}_{inv}(\mathbf{q}) = \begin{pmatrix} \rho u_j \\ \rho u_i u_j + \delta_{ij} p \\ u_j (E + p) \end{pmatrix}, \text{ and } \mathbf{f}_{vis}(\mathbf{q}, \nabla \mathbf{q}) = \begin{pmatrix} 0 \\ \tau_{ij} \\ u_i \tau_{ij} + K_j \end{pmatrix}, \quad (2)$$

where

$$\tau_{ij} = \left(\frac{u_i}{x_j} + \frac{u_j}{x_i} \right) - \frac{2}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij}, \text{ and } K_j = k \frac{\partial T}{\partial x_j}. \quad (3)$$

In this study, $k = \mu C_p / Pr$. The viscosity μ is treated as a constant and $Pr = 0.72$.

3 Spatial discretization

If we transfer the Navier-Stokes equations from the physical domain (x, y, z) to the computational domain (ξ, η, ζ) , Eq. (1) can be expressed as

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial \xi} + \frac{\partial \mathbf{G}}{\partial \eta} + \frac{\partial \mathbf{H}}{\partial \zeta} = 0, \quad (4)$$

where

$$\begin{cases} \mathbf{Q} = |\mathbf{J}| \mathbf{q}, \\ \mathbf{F} = |\mathbf{J}| (\mathbf{f} \xi_x + \mathbf{g} \xi_y + \mathbf{h} \xi_z), \\ \mathbf{G} = |\mathbf{J}| (\mathbf{f} \eta_x + \mathbf{g} \eta_y + \mathbf{h} \eta_z), \\ \mathbf{H} = |\mathbf{J}| (\mathbf{f} \zeta_x + \mathbf{g} \zeta_y + \mathbf{h} \zeta_z), \end{cases} \quad (5)$$

and

$$\mathbf{J} = \frac{\partial(x, y, z)}{\partial(\xi, \eta, \zeta)}, \text{ and } |\mathbf{J}| = \det(\mathbf{J}). \quad (6)$$

The flux polynomial reconstructed from the FR/CPR method consists of two parts, one of which is the local flux polynomial and the other is the correction polynomial. On solving Eq. (4), the reconstructed polynomials $\tilde{\mathbf{F}}$, $\tilde{\mathbf{G}}$ and $\tilde{\mathbf{H}}$ of \mathbf{F} , \mathbf{G} and \mathbf{H} can be expressed as

$$\begin{cases} \tilde{\mathbf{F}} = \mathbf{F}^l + \mathbf{F}^c, \\ \tilde{\mathbf{G}} = \mathbf{G}^l + \mathbf{G}^c, \\ \tilde{\mathbf{H}} = \mathbf{H}^l + \mathbf{H}^c, \end{cases} \quad (7)$$

where the subscript ‘l’ stands for the local flux and ‘c’ stands for the correction flux. Consequently, Eq. (1) can be written as

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{f}^l}{\partial x} + \frac{\partial \mathbf{g}^l}{\partial y} + \frac{\partial \mathbf{h}^l}{\partial z} + \frac{1}{|\mathbf{J}|} \left(\frac{\partial \mathbf{F}^c}{\partial \xi} + \frac{\partial \mathbf{G}^c}{\partial \eta} + \frac{\partial \mathbf{H}^c}{\partial \zeta} \right) = 0 \quad (8)$$

For hexahedral elements, \mathbf{F}^c , \mathbf{G}^c and \mathbf{H}^c can be explicitly expressed as

$$\begin{cases} \mathbf{F}^c(\xi, \eta, \zeta) = (\tilde{\mathbf{F}}(-1, \eta, \zeta) - \mathbf{F}^l(-1, \eta, \zeta))g_L(\xi) + (\tilde{\mathbf{F}}(1, \eta, \zeta) - \mathbf{F}^l(1, \eta, \zeta))g_R(\xi), \\ \mathbf{G}^c(\xi, \eta, \zeta) = (\tilde{\mathbf{G}}(\xi, -1, \zeta) - \mathbf{G}^l(\xi, -1, \zeta))g_L(\eta) + (\tilde{\mathbf{G}}(\xi, 1, \zeta) - \mathbf{G}^l(\xi, 1, \zeta))g_R(\eta), \\ \mathbf{H}^c(\xi, \eta, \zeta) = (\tilde{\mathbf{H}}(\xi, \eta, -1) - \mathbf{H}^l(\xi, \eta, -1))g_L(\zeta) + (\tilde{\mathbf{H}}(\xi, \eta, 1) - \mathbf{H}^l(\xi, \eta, 1))g_R(\zeta), \end{cases} \quad (9)$$

where $g_{L/R}$ are the correction polynomials. In this study, we employ the Radau polynomials to recover the nodal FR-DG method. $\tilde{\mathbf{F}}$, $\tilde{\mathbf{G}}$ and $\tilde{\mathbf{H}}$ at element interfaces are referred as numerical fluxes \mathbf{F}^{num} , \mathbf{G}^{num} and \mathbf{H}^{num} . The inviscid common fluxes can be obtained from approximate Riemann solvers. In this study, the Rusanov solver is used to calculate the common fluxes at the cell interfaces as

$$\mathbf{f}_{\mathbf{n},inv}^{com} = \frac{\mathbf{f}_{\mathbf{n},inv}^+ + \mathbf{f}_{\mathbf{n},inv}^-}{2} - |\lambda|_{max} \frac{\mathbf{q}^- - \mathbf{q}^+}{2}, \quad (10)$$

where superscripts ‘+’ and ‘-’ denote the left of right side of the current interface and \mathbf{n} is the normal direction. Numerical common fluxes can be obtained as

$$\begin{cases} \mathbf{F}^{num} = |\mathbf{J}| |\nabla \xi| \mathbf{f}_{\mathbf{n}}^{com} \text{sign}(\mathbf{n} \cdot \nabla \xi), \\ \mathbf{G}^{num} = |\mathbf{J}| |\nabla \eta| \mathbf{f}_{\mathbf{n}}^{com} \text{sign}(\mathbf{n} \cdot \nabla \eta), \\ \mathbf{H}^{num} = |\mathbf{J}| |\nabla \zeta| \mathbf{f}_{\mathbf{n}}^{com} \text{sign}(\mathbf{n} \cdot \nabla \zeta). \end{cases} \quad (11)$$

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

$$\mathbf{q}^{com} = \frac{\mathbf{q}^+ + \mathbf{q}^-}{2}. \quad (12)$$

The common gradient is computed as

$$\nabla \mathbf{q}^{com} = \frac{\nabla \mathbf{q}^+ + \mathbf{r}^+ + \nabla \mathbf{q}^- + \mathbf{r}^-}{2}, \quad (13)$$

where \mathbf{r}^+ and \mathbf{r}^- are the corrections to the gradients on the interface. For the hexahedral element, the correction terms are defined as [26]

$$\mathbf{r} = \gamma(\mathbf{q}^{com} - \mathbf{q})\mathbf{n}, \quad \text{and } \gamma = |\nabla \varpi| g'(\varpi) \text{sign}(\mathbf{n} \cdot \nabla \varpi), \quad (14)$$

where $\varpi \in \{\xi, \eta, \zeta\}$ and ϖ is -1 or 1 depending on the positions of the current interface in the left and right side elements. In this study, $g(\pm 1) = \pm(p+1)(p+2)/2$ to stabilize the method.

The configuration procedure of the FR/CPR method reveals that the spatial discretization is compact and only needs information from all the neighbor elements for the spatial derivative approximations. This compact reconstruction procedure is well-suited for high performance computing.

4 Time integration

If we simplify Eq. (8) as

$$\frac{\partial \mathbf{q}}{\partial t} = \mathbf{R}(\mathbf{q}), \quad (15)$$

the general form of ROIRK methods from time step n to $n + 1$ can be written as

$$\begin{cases} \mathbf{q}^{n+1} = \mathbf{q}^n + \sum_{j=1}^s m_j Y_j, \\ \left(\frac{\mathbf{I}}{\omega\Delta t} - \frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right)^n Y_i = \mathbf{R}\left(\mathbf{q}^n + \sum_{j=1}^{i-1} a_{ij} Y_j\right) + \sum_{j=1}^{i-1} \frac{c_{ij}}{\Delta t} Y_j, \quad i = 1, 2, \dots, s, \end{cases} \quad (16)$$

where s is the number of stages. Details about the coefficients of a series of ROIRK methods can be found in Ref. [20]. The Jacobian matrix $\frac{\partial \mathbf{R}}{\partial \mathbf{q}}$ only needs to be updated at the beginning of each time step.

In this study, we use the complex-step derivative approximation [24] to calculate the Jacobi matrix. There are multiple options to calculate the Jacobi matrix, among which are the analytical derivation, automatic differentiation and finite difference approach. The finite difference approach is quite popular due to its easy implementation. However, the lack of accuracy definitely will degenerate the accuracy property of the Rosenbrock-type Runge-Kutta methods. And it will make the linear solver hard to converge. Though the analytical approach can guarantee the accuracy, when applied to the Navier-Stokes equations, the derivation can be cumbersome. Many automatic differentiation tools are available to automatically generate the code for Jacobi evaluation when the source code of the $\mathbf{R}(\mathbf{q})$ is supplied, such as TAPENADE [27] or TAMC [28]. However, the output code could be lengthy and hard for debugging and maintenance. The complex-step derivative approximation follows the following Taylor expansion,

$$f(x + ih) = f(x) + ihf'(x) - h^2 \frac{f''(x)}{2!} - ih^3 \frac{f'''(x)}{3!} + \dots, \quad (17)$$

where $i^2 = -1$. Taking the imaginary parts of both sides of the Taylor series and dividing them by h gives

$$f'(x) = \frac{\text{Im}(f(x + ih))}{h} + O(h^2). \quad (18)$$

This approximation is accurate to $O(h^2)$. In double-precision simulations, when $h = 10^{-8}$, the complex-step derivative approximation will be accurate to machine zero. This approximation offers an easy implementation that one can substitute all the variables in the CFD code related to the flow with corresponding complex numbers. And a small disturbance can be simply added to the imaginary part of the conservative variables when calculating the Jacobi matrix.

Bassi et al. have investigated the accuracy performance of ROIRK methods up to sixth order with DG methods on solving both compressible and incompressible flows in Ref. [20]. In this study, we focus on the parallel implementation and scalability study of the ROIRK methods. In this study, the two-stage, second-order ROIRK method (RO2-2) [14] method, the three-stage, third-order ROIRK (ROS3P) [15] and the six-stage, fourth-order ROIRK method (RODASP [16]) are investigated. A notation ROIRK m - n , where ' m ' stands for the order of accuracy and ' n ' stands for the number of stages, is used in this study to avoid confusions.

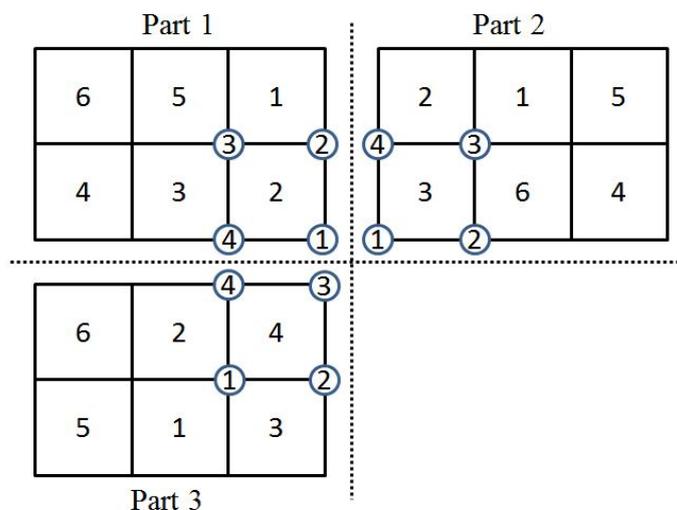


Figure 1: Illustration of the connectivities of different parts of the 2D unstructured mesh.

5 Details on the Parallel Implementation

In this section, the parallel implementation is discussed in detail. For the pre-processing, specifically, the `METIS_PartMeshDual` function in METIS [23], is used to partition the mesh based on the partitioning of the dual graph of the mesh. In the present study, we partition the mesh of N elements into M parts and distribute each part to one process. Each part has N_m cells. $\mathbf{q}_{i,m}$ and $\mathbf{R}_{i,m}$ are employed to denote the conservative variable vector and r.h.s. vector of the i -th cell on partition m . Herein, $i = 1, 2, \dots, N_m$ and $m = 1, 2, \dots, M$.

After partitioning the mesh, connectivities need to be built up among different partitions. The element faces shared by different parts can be regarded as special types of boundary conditions. Note that a pair of periodic faces can also be shared by two parts. A simple 2D illustration of the connectivities has been presented in Figure 1. In Figure 1, Part 1 is geometrically connected to Part 2 and Part 3. Two different types of shared-faces are involved, i.e., two faces are shared by Part 1 and Part 2, and three faces are shared by Part 1 and Part 3. Taking the Element 2 in Part 1 as an example, the first face (from node 1 to 2) is the same face as the fourth face (from node 4 to node 1) in Element 3 in Part 2 except that the orientation is different. For FR/CPR methods, it is critical that the orientations of these faces must match when calculating common fluxes on the flux points.

The PETSc library is employed for storing the sparse Jacobian matrix. In PETSc library, the Jacobian matrix \mathbf{J}_m is distributed into different processes by splitting the rows of the full matrix \mathbf{J} [25]. The distributed Jacobian matrix on the m -th process can

be explicitly written as

$$\mathbf{J}_m = \begin{pmatrix} \frac{\partial \mathbf{R}_{1,m}}{\partial \mathbf{q}_1} & \frac{\partial \mathbf{R}_{1,m}}{\partial \mathbf{q}_2} & \cdots & \frac{\partial \mathbf{R}_{1,m}}{\partial \mathbf{q}_j} & \cdots & \frac{\partial \mathbf{R}_{1,m}}{\partial \mathbf{q}_N} \\ \frac{\partial \mathbf{R}_{2,m}}{\partial \mathbf{q}_1} & \frac{\partial \mathbf{R}_{2,m}}{\partial \mathbf{q}_2} & \cdots & \frac{\partial \mathbf{R}_{2,m}}{\partial \mathbf{q}_j} & \cdots & \frac{\partial \mathbf{R}_{2,m}}{\partial \mathbf{q}_N} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{R}_{N_m,m}}{\partial \mathbf{q}_1} & \frac{\partial \mathbf{R}_{N_m,m}}{\partial \mathbf{q}_2} & \cdots & \frac{\partial \mathbf{R}_{N_m,m}}{\partial \mathbf{q}_j} & \cdots & \frac{\partial \mathbf{R}_{N_m,m}}{\partial \mathbf{q}_N} \end{pmatrix} \quad (19)$$

Recall Eqs. (10), (12) and (13), in order to calculate the common inviscid and viscous fluxes, information from neighbor elements will be needed. Therefore, in the procedure of calculating $\mathbf{R}_{i,m}$ and $\frac{\partial \mathbf{R}_{i,m}}{\partial \mathbf{q}_j}$, communications among different processes are needed when \mathbf{q}_j is not on the current process m . Note that when the element of global index j is neither the current local element i nor the neighbor of this element, $\frac{\partial \mathbf{R}_{i,m}}{\partial \mathbf{q}_j} = 0$. If a p degree polynomial reconstruction is used in the FR/CPR method, for a hexahedral element, the number of degrees of freedom is $n_{dof} = n_v(p+1)^3$, where $n_v = 5$ is the number of conservative variables. And the dimensions of a non-zero $\frac{\partial \mathbf{R}_{i,m}}{\partial \mathbf{q}_j}$ block are $n_{dof} \times n_{dof}$. Therefore, to evaluate $\frac{\partial \mathbf{R}_{i,m}}{\partial \mathbf{q}_j}$ with the complex-step derivative approximation, adding a disturbance to the corresponding entry in \mathbf{q}_j will be done n_{dof} times. The most efficient way in terms of communication is to send all the \mathbf{q}_j 's of those elements that share faces with the local mesh from other processes to the local process. And `MPI_Isend` and `MPI_Irecv` are employed for message passing.

The GMRES solver in PETSc library has been parallelized as well as the block Jacobi preconditioner and additive Schwarz preconditioner. Many literatures have discussed the scalability of GMRES solver with different parallel preconditioners [29]. In this study, we simply employ block Jacobi preconditioner, which is equivalent to the additive Schwarz preconditioner with non-overlapping, as the preconditioner. Each process possesses only one block and ILU(0) is adopted as the local preconditioner on each block. For this study, the full Jacobi matrix is stored in memory as well as the corresponding preconditioner matrix. The current implementation has a significant memory usage when the problem size is big. The Rosenbrock-Krylov methods [18] and Rosenbrock-Wanner methods can be good alternatives since they have the advantage that the Jacobi can be non-exactly approximated. This will be considered in future work. Tips about efficiently using PETSc can also be found in Ref. [29].

6 Numerical Results

All simulations are carried out on the HPCF2013 portion of the maya cluster, the UMBC High Performance Computing Facility (HPCF). HPCF2013 has 72 nodes. Every node has two Intel E5-2650v2 Ivy Bridge (2.6 GHz, 20 MB cache) processors with eight cores apiece, i.e., 16 cores per node. The speedup S_{np} is defined as

$$S_{np} = T_{np_{ref}}/T_{np}, \quad (20)$$

where $T_{np_{ref}}$ is the CPU time when np_{ref} processes are used and T_{np} is the CPU time when np processes are used. The observed efficiency E_{np} is defined as

$$E_{np} = \frac{S_{np}}{np/np_{ref}}. \quad (21)$$

Normally, the scalability study will be conducted with $np_{ref} = 1$. However, the current implementation is matrix-based. Thus, for bigger problems, the memory on one node will not be enough to carry out the simulation. In order to investigate the scalability performance of the ROIRK methods on solving relative bigger problems, $np_{ref} > 1$ will be employed.

A notice is that the memory allocation for a big matrix, even though it is a sparse matrix, takes a long time. Since the non-zero pattern of the Jacobi matrix and the preconditioner matrix will not change, no allocation will be executed once the matrices are assigned to certain memory addresses. In order to avoid its effect on timing for the parallel performance study, the code will first march one step with $\Delta t = 0$.

6.1 Isentropic Vortex Propagation

The isentropic vortex propagation case depicts the superposition of an inviscid uniform flow and an irrotational vortex. The vortex can be regarded as a perturbation added into the uniform flow. The free stream flow is of $(\rho, u, v, w, Ma) = (1, 1, 1, 0, 0.5)$ and $R = 1.0$ for this case. The perturbation is defined as [20]

$$\begin{cases} \delta u = -\frac{\alpha}{2\pi}(y - y_0)e^{\phi(1-r^2)}, \\ \delta v = \frac{\alpha}{2\pi}(x - x_0)e^{\phi(1-r^2)}, \\ \delta w = 0, \\ \delta T = -\frac{\alpha^2(\gamma-1)}{16\phi\gamma\pi^2}(y - y_0)e^{2\phi(1-r^2)}, \\ dS = 0, \end{cases} \quad (22)$$

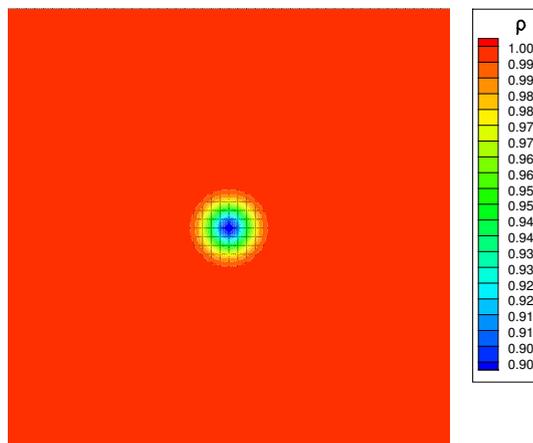
where $\phi = \frac{1}{2}$ and $\alpha = 5$ are parameters that define the vortex strength. $r = (x - x_0)^2 + (y - y_0)^2$ is the distance to the center of the vortex $(x_0, y_0) = (0, 0)$ at $t = 0$. The periodic domain is defined in $[-10, 10] \times [-10, 10]$. Periodic boundary conditions are used for all four boundaries. For this case, the convergence criteria of the GMRES solver is $tol_{res} = 10^{-10}$ and $n_{restart} = 60$.

We use the P^4 FR/CPR scheme on a $50 \times 50 \times 1$ uniform mesh to guarantee that the error is dominated by the time integration. The simulation stops at $t = T$, where $T = 20$ is the period. The density contour at $t = T$ of the ROIRK4-6 method is illustrated in Figure 2. The convergence study of the time integrations are presented in Table 1. Numerical results have shown that the convergence orders of ROIRK2-2 and ROIRK4-6 agree well with theoretical values in general. However, a slight order reduction of the ROIRK3-3 is observed.

Scalability studies have been conducted for all three types of ROIRK methods. The CPU time of $np_{ref} = 1$ is used as the reference value. A $64 \times 64 \times 1$ mesh and the

Table 1: The convergence study for ROIRK methods.

Δt	$E_{L_2}(\rho)$	order	$E_{L_2}(u)$	order	optimal
ROIRK2-2					
T/100	1.2754×10^{-3}		1.6299×10^{-2}		2
T/200	3.2468×10^{-4}	1.97	4.1619×10^{-3}	1.97	2
T/400	8.1380×10^{-5}	2.00	1.0452×10^{-3}	1.99	2
ROIRK3-3					
T/100	1.2009×10^{-3}		1.2723×10^{-2}		3
T/200	2.0399×10^{-4}	2.56	2.3046×10^{-3}	2.46	3
T/400	2.9528×10^{-5}	2.79	3.2167×10^{-4}	2.84	3
ROIRK4-6					
T/100	1.1154×10^{-5}		1.0505×10^{-4}		4
T/200	7.0437×10^{-7}	3.99	6.6557×10^{-6}	3.98	4
T/400	5.1761×10^{-8}	3.77	4.5476×10^{-7}	3.87	4

Figure 2: The density contour at $t = T$ using ROIRK4-6 with the P^4 FR/CPR method for the vortex propagation case.

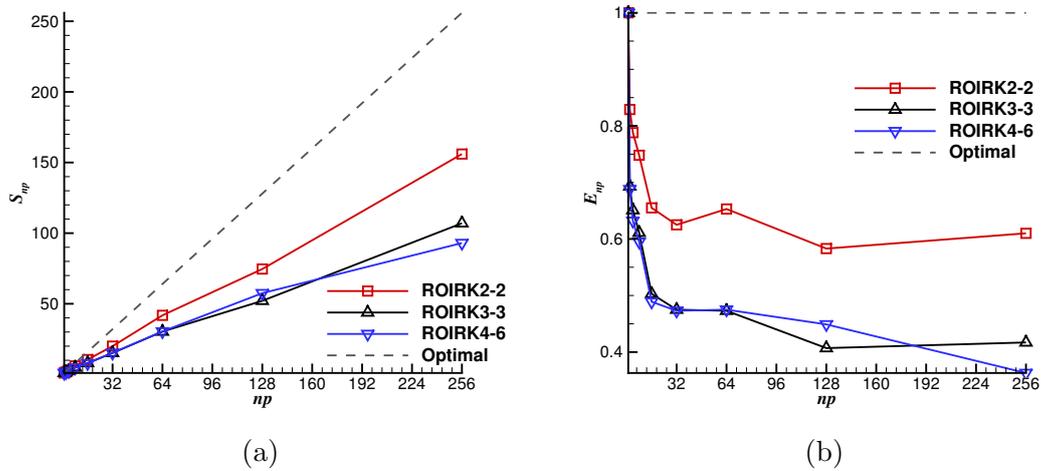


Figure 3: (a) The speedup S_p and (b) observed efficiency E_p results of ROIRK2-2, ROIRK3-3 and ROIRK4-6 with the P^2 FR/CPR method on solving the vortex propagation on the $50 \times 50 \times 1$ mesh.

P^2 FR/CPR method is employed for these studies. For this case, $\Delta t = T/100$ and we only time one step due to the fact the simulation is slow with only one process. The speedup and observed efficiency of different ROIRK methods are illustrated in Figure 3. The average number of GMRES iterations per stage is illustrated in Figure 4. As the number of partitions increase, more iterations will be needed for the linear solver to converge. This is due to the block Jacobi preconditioner. For the ROIRK2-2 method, the observed efficiency is above 0.6. Generally speaking, as the number of stage of the ROIRK methods increases, since linear solver itself is not perfectly scalable, the scalability of the ROIRK methods will decrease according to the stage number ratio compared to ROIRK2-2. However, the unexpected poor scalability of ROIRK3-3 is due to the factor that ROIRK3-3 needs significantly more iterations for the GMRES solver to converge.

6.2 Laminar Flow Past a Cylinder

In this section, the laminar flow of $Ma = 0.2$ and $Re = 100$ past a circular cylinder is tested to study the parallel performance of different ROIRK methods. The computational domain is $[x, y, z] \in [-16, 100] \times [-16, 16] \times [0, 1]$. An illustration of the domain and mesh is presented in Figure 5. The mesh employed in this study has a 11000 elements in total. Specifically, there are 200 elements in the circumferential direction of the cylinder and 50 elements in the normal direction and only on element along the span-wise direction. The P^2 elements are employed to represent the curvature of the wall boundaries. The P^3 FR/CPR scheme is used as the spatial discretization. The front and back side of the domain are treated as periodic boundaries. The top and bot-

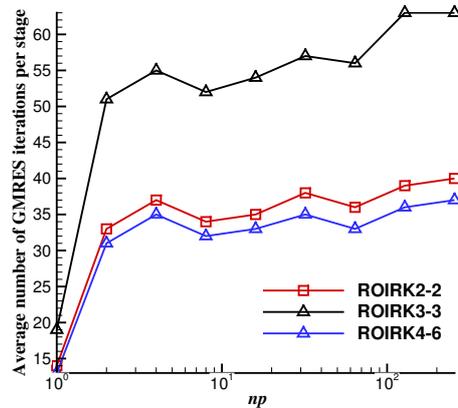


Figure 4: The average number of GMRES iterations per stage of ROIRK2-2, ROIRK3-3 and ROIRK4-6 on solving the vortex propagation on the $64 \times 64 \times 1$ mesh with the P^2 FR/CPR method.

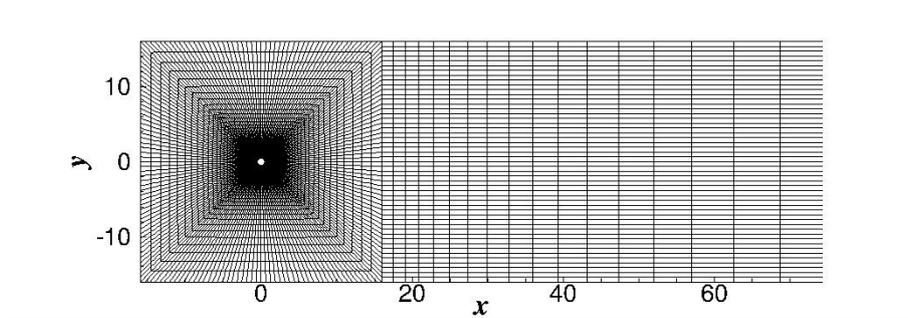


Figure 5: An illustration of the domain and mesh of the laminar flow past a cylinder.

tom of the domain are symmetric boundaries. The left and right boundary are loosely enforce as $(\rho_\infty, u_\infty, 0, 0, p_\infty)$ with Riemann solvers. The adiabatic wall boundary condition is applied to the wall surface. The converging tolerance of the GMRES solver is $tol_{res} = 10^{-5}$ and $n_{restart} = 60$. A simulation of $t = 200$ is conducted as a validation of the code using the ROIRK2-2 method. The time step is set as $\Delta t = 0.05$. An instance of the vorticity field of the vortex shedding is presented in Figure 6. The histories of C_l and C_d are illustrated in Figure 7. The average drag coefficient $\overline{C_d} = 1.377$, root mean square (RMS) of the lift coefficient $C_{l,rms} = 0.235$ and Strouhal number $St = 0.166$, which are slightly bigger than those in Ref. [30] for the 2D problem. This difference is due to the insufficient spatial resolution in span-wise direction. However, it will not substantially change the scalability study.

The scalability study for this case starts with $np_{ref} = 64$. And we time 10 time steps. The S_{np} and E_{np} are illustrated in Figure 8. The average numbers of GMRES iterations

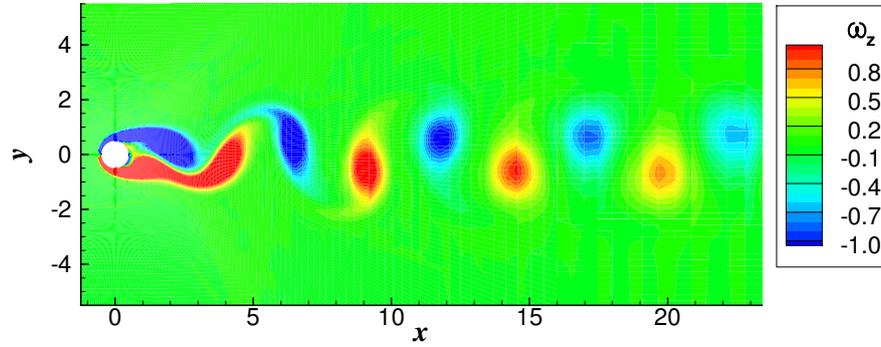


Figure 6: An instance of the vortex shedding of the laminar flow of $Ma = 0.2$ and $Re = 100$ past a cylinder.

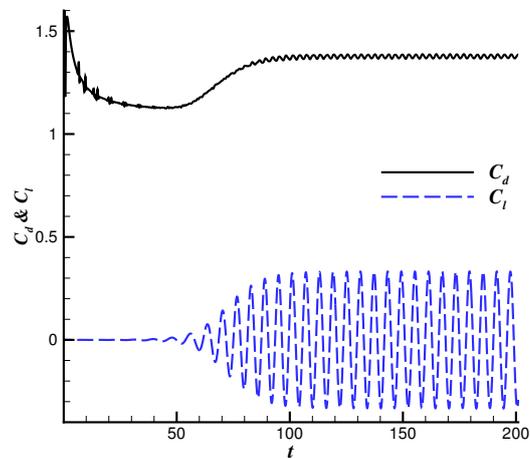


Figure 7: The histories of C_d and C_l of the laminar flow of $Ma = 0.2$ and $Re = 100$ past a cylinder.

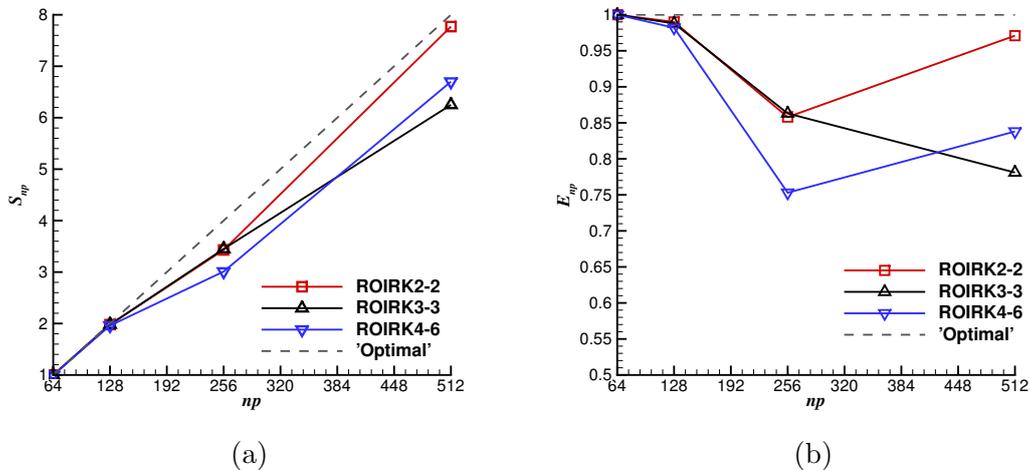


Figure 8: (a) The speedup S_p and (b) observed efficiency E_p results of ROIRK2-2, ROIRK3-3 and ROIRK4-6 on solving laminar flow past the cylinder.

per stage of all three ROIRK methods are presented in Figure 9. For a larger problem, the scalability of ROIRK methods is better compared to the vortex propagation problem. And the ROIRK3-3 method on solving the viscous problem also requires more iterations per stage than ROIRK2-2 and ROIRK4-6. The parallel efficiency of ROIRK3-3 is even worse than ROIRK4-6 when $np = 512$.

7 Conclusions and Future Work

High-order ROIRK methods have been implemented for the parallel simulation of the 3D Navier-Stokes equations spatially discretized with the FR/CPR formulation. In this study, ROIRK methods have demonstrated high order of accuracy in terms of temporal integration. A small size inviscid problem (the vortex propagation) and a relative large size viscous (the laminar flow past a cylinder) problem have been tested to investigate the scalability performance of the ROIRK methods. Generally speaking, ROIRK2-2 has better scalability than ROIRK3-3 and ROIRK4-6. The linear system results from the ROIRK3-3 method with FR/CPR method is stiffer than those from ROIRK2-2 and ROIRK4-6. As a consequence, more iterations are needed for the GMRES to solve the linear systems per stage. Hence, worse scalability of ROIRK3-3 is observed when considering the number of stages. A matrix-based parallel implementation of the ROIRK methods has been employed in this study. However, the significant requirement on the memory hinders the application of this approach for solving realistic engineering problems involving turbulence. Thus, a matrix-free implementation of Rosenbrock-Wanner or Rosenbrock-Krylov methods will be more preferable rather than traditional Rosenbrock-type methods for future research. Also, the Jacobi preconditioner will suffer from lower convergence speed when the number of blocks increases. The additive Schwartz method

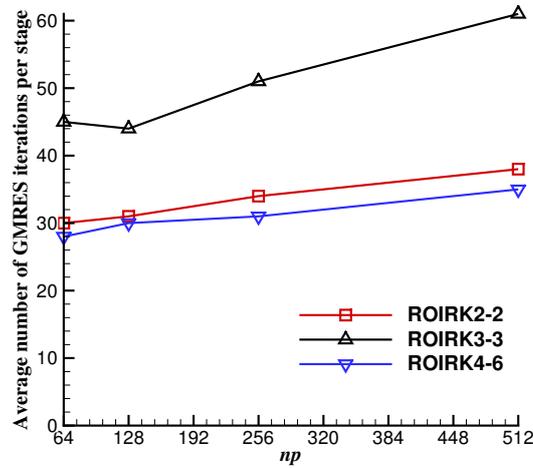


Figure 9: The average number of GMRES iterations per stage of ROIRK2-2, ROIRK3-3 and ROIRK4-6 on solving laminar flow past the cylinder on a 11000-element mesh with the 4th order FR/CPR method.

with overlapping can be an alternative.

Acknowledgment

The authors gratefully acknowledge the support of the Office of Naval Research through the award N00014-16-1-2735, and the National Science Foundation through the award OAC-1726023.

References

- [1] 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.
- [2] 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.
- [3] 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.
- [4] 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.
- [5] J. Romero, K. Asthana and A. Jameson, “A simplified formulation of the flux reconstruction method,” *Journal of Scientific Computing*, vol. 67, pp. 351–372, 2016.
- [6] L. Wang and M. Yu, “Compact direct flux reconstruction for conservation laws,” *Journal of Scientific Computing*, pp. 1–23, 2017.
- [7] 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.
- [8] 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*.
- [9] 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.
- [10] 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.
- [11] A. Uranga, P.-O. Persson, 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, no. 1-5, pp. 232–261, 2011.

- [12] G. Wanner and E. Hairer, “Solving ordinary differential equations ii,” *Stiff and Differential-Algebraic Problems*, 1991.
- [13] A. Sarshar, P. Tranquilli, B. Pickering, A. McCall, A. Sandu, and C. J. Roy, “A numerical investigation of matrix-free implicit time-stepping methods for large cfd simulations,” *Computers & Fluids*, vol. 159, pp. 53–63, 2017.
- [14] 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.
- [15] 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.
- [16] G. Steinebach, “Order-reduction of row-methods for daes and method of lines applications,” 1995.
- [17] 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.
- [18] 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.
- [19] P. Tranquilli, S. R. Glandon, A. Sarshar, and A. Sandu, “Analytical jacobian-vector products for the matrix-free time integration of partial differential equations,” *Journal of Computational and Applied Mathematics*, vol. 310, pp. 213–223, 2017.
- [20] 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.
- [21] 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.
- [22] X. D. Liu, Y. D. Xia, H. Luo, and L. J. 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, pp. 1016–1044, 2016.
- [23] G. Karypis and V. Kumar, “A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs,” *SIAM Journal on Scientific Computing*, vol. 20, pp. 359–392, 1999.

- [24] J. R. Martins, P. Sturdza, and J. J. Alonso, “The complex-step derivative approximation,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 29, no. 3, pp. 245–262, 2003.
- [25] 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.
- [26] 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.
- [27] L. Hascoët and V. Pascual, “The Tapenade automatic differentiation tool: Principles, model, and specification,” *ACM Transactions on Mathematical Software*, vol. 39, no. 3, pp. 20:1–20:43, 2013.
- [28] R. Giering and T. Kaminski, “Recipes for adjoint code construction,” *ACM Transactions on Mathematical Software*, vol. 24, no. 4, pp. 437–474, 1998.
- [29] W. Gropp, D. Keyes, L. C. McInnes, and M. D. Tidriri, “Globalized newton-krylov-schwarz algorithms and software for parallel implicit cfd,” *The International Journal of High Performance Computing Applications*, vol. 14, no. 2, pp. 102–136, 2000.
- [30] L. Wang and M. Yu, “Compact direct flux reconstruction for the navier-stokes equations on dynamic meshes,” in *23rd AIAA Computational Fluid Dynamics Conference*, p. 3098, 2017.