Residual-Based Time-Step Control for High-Order Discretizations

Krzysztof J. Fidkowski*
University of Michigan, Ann Arbor, MI, 48188

This paper presents a study of temporal errors, as well as a time-step control strategy for their reduction, in high-order spatial discretizations of convection-dominated flow equations. The discontinuous Galerkin finite-element method serves as the spatial discretization, and it is combined with implicit, semi-discrete time-marching schemes. Efficiency of the schemes, measured in terms of the number of implicit system solutions for a given level of accuracy, is compared for different spatial orders and mesh sizes. For uniform time stepping, certain hybrid multi-step/stage schemes can be more efficient than popular diagonally-implicit Runge-Kutta methods. A residual-based adaptive time-step control strategy is also presented for balancing spatial and temporal errors in each time step, with a focus on performance on spatially-adapted meshes. The methods are tested on an unsteady manufactured solution for scalar advection-diffusion, and for the Navier-Stokes equations on two problems with moving geometry and mesh.

I. Introduction

In the area of computational fluid dynamics, high-order spatial discretizations [1], particularly when combined with output-based mesh adaptation [2], have yielded a robust solution approach for steady-state problems. Discretization errors can be estimated and reduced through mesh-adaptive techniques. Less effort has been devoted to unsteady problems, for which high-order spatial methods are paired with high-order temporal schemes. In the presence of non-smooth temporal solutions or output definitions, which are common in practical engineering problems, high-order accuracy in time can be difficult to attain. In addition, high-order systems are often stiff and require implicit time-integration methods, which allow for a large time step but also burden the user with selecting the appropriate time step. As a result, many practical unsteady simulations may be done with a poor choice of time step: one that is too large leads to high temporal errors that drown out the benefits of high-order spatial accuracy, whereas a time step that is too small wastes computational resources. Typically, time-step convergence studies are performed to ascertain the appropriate time step, at a cost of multiple unsteady simulations.

Prior to choosing the time step, there is also the question of what time marching method to use. Classic explicit methods [3] become inefficient with increasing stiffness, which is exacerbated by high spatial order, adapted meshes, turbulence modeling, anisotropic meshes, etc. As a result, much attention has been given to implicit time integration methods in such cases, including highly-accurate but expensive variational methods [4, 5] and more common semi-discrete methods [6, 7]. The latter are relatively simple to implement starting from an implicit steady solver, and they are the focus of this work.

Many semi-discrete methods have been developed for integrating general stiff systems of ordinary differential equations, not necessarily those arising from high-order spatial discretizations. These include multi-step methods, which are inexpensive but of limited stability at high order, multi-stage methods, which do not require a history of multiple steps but often need many stages per time step, and hybrid methods that blend the two.

Desirable characteristics of a time-integration scheme include a high order of accuracy, low cost, A-stability, and often L-stability [6]. The stability properties are important for high spatial-order discretizations of convection-dominated flows, which may have a combination of eigenvalues with large-magnitude negative real parts and ones close to the imaginary axis. When using adaptive time stepping, another consideration is the requirement of uniform time steps, characteristic of multi-step methods, which must be replaced by another method at the initial time or when the time step changes.

In this work, we investigate the accuracy and efficiency of various semi-discrete time-marching schemes for high-order spatial discretizations of advection-dominated flows. We look at the temporal convergence properties of the

*Professor, Department of Aerospace Engineering, AIAA Associate Fellow.

methods, as measured by errors in the flowfield and scalar outputs, and the cost as measured by the number of nonlinear solutions required to attain a given error. We also present a time-step control strategy that strives to balance the temporal and spatial errors, i.e. to hit the "sweet-spot" for unsteady simulations in which just the right amount of computational effort is expended for the time integration in light of the spatial errors. To this end, we do not consider output-based methods that require adjoints, which can be very effective [8–15] but which require a high implementation and solution cost for unsteady problems that makes them out of reach for many codes.

The remainder of this paper presents the discretizations studied, with emphasis on a self-contained exposition of the time-marching schemes, and the time-step control strategy that uses error estimates obtained from fine-space residual evaluations. Results on prototypical problems, including the Navier-Stokes equations on deforming domains, compare the accuracy and efficiency of the time-marching schemes and demonstrate the performance of the time-step control strategy.

II. Discretization

This section presents the spatial and temporal discretizations used in this study. Similar results would likely be attained with different high-order spatial discretizations, while the temporal scheme discussion is meant to be a comprehensive exposition of various popular methods.

A. Spatial

We consider discretizations of differential equations in conservation form,

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{H}}(\mathbf{u}, \nabla \mathbf{u}) + \mathbf{S}(\mathbf{u}) = \mathbf{0}, \tag{1}$$

where $\mathbf{u}(\vec{x}, t) \in \mathbb{R}^s$ is the conservative state vector, s is the state rank, $\vec{\mathbf{H}}(\mathbf{u}, \nabla \mathbf{u}) = \vec{\mathbf{F}}(\mathbf{u}) + \vec{\mathbf{G}}(\mathbf{u}, \nabla \mathbf{u})$ is the combined inviscid and viscous flux, and $\mathbf{S}(\mathbf{u})$ is a source term, active in this work when applying the method of manufactured solutions.

To discretize Eqn. 1 in space, we use the discontinuous Galerkin finite-element method [16]. The computational domain Ω is divided into N_e elements, Ω_e, in a non-overlapping tessellation T_h. Inside element Ω_e, the state components are approximated by polynomials of order p, with no continuity constraints between elements. Formally, we write: $\mathbf{u}_h \in \boldsymbol{\mathcal{V}}_h = [\mathcal{V}_h]^s$, where $\mathcal{V}_h = \left\{ u \in L_2(\Omega) : u|_{\Omega_e} \in \mathcal{P}^p \ \forall \Omega_e \in T_h \right\}$, and $\mathcal{P}^p$ denotes polynomials of order p on the reference space of element Ω_e.

The discontinuous Galerkin weak form of Eqn. 1 is obtained by multiplying the equation by test functions, $\mathbf{w}_h$, in the same approximation space, integrating by parts, and coupling elements via single-valued fluxes that are functions of the states on the two adjacent elements. The weak form then reads: find $\mathbf{u}_h \in \boldsymbol{\mathcal{V}}_h$ such that $\forall \mathbf{w}_h \in \boldsymbol{\mathcal{V}}_h$,

$$-\int_{\Omega_e} \nabla \mathbf{w}_h^T \cdot \vec{\mathbf{H}}(\mathbf{u}_h, \nabla \mathbf{u}_h)\, d\Omega + \int_{\partial\Omega_e} \mathbf{w}_h^T \widehat{\mathbf{H}} \cdot \vec{n}\, ds - \int_{\partial\Omega_e} \partial_i \mathbf{w}_h^{+T} \mathbf{K}_{ij}^{+} \left(\mathbf{u}_h^{+} - \widehat{\mathbf{u}}_h \right) n_j\, ds + \int_{\Omega_e} \mathbf{w}_h^T \mathbf{S}(\mathbf{u}_h) = 0, \tag{2}$$

where $(\cdot)^T$ denotes transpose, $\widehat{\mathbf{u}}_h = (\mathbf{u}_h^{+} + \mathbf{u}_h^{-})/2$, $\mathbf{K}_{ij}$ is the diffusivity tensor arising from $\mathbf{G}_i = \mathbf{K}_{ij}(\mathbf{u})\partial_j \mathbf{u}$, and on the element boundary $\partial\Omega_e$, $(\cdot)^{+}$, $(\cdot)^{-}$ denote quantities taken from the element or its neighbor (or boundary condition), respectively. For the normal flux, $\widehat{\mathbf{H}} \cdot \vec{n}$, we use the Roe-approximate Riemann solver [17] and the second form of Bassi and Rebay (BR2) [18] for the viscous flux. Additional details of the spatial discretization can be found in our previous work [19]. Choosing a basis for the test and trial spaces yields a system of nonlinear equations,

$$\mathbf{M} \frac{d\mathbf{U}}{dt} + \mathbf{R}(\mathbf{U}) = \mathbf{0}, \tag{3}$$

where $\mathbf{U} \in \mathbb{R}^N$ is the discrete state vector, $\mathbf{R}(\mathbf{U}) \in \mathbb{R}^N$ is the spatial residual vector, $\mathbf{M} \in \mathbb{R}^{N \times N}$ is the block-sparse mass matrix, and N is the total number of spatial degrees of freedom.

B. Temporal

To advance Eqn. 3 forward in time, we consider multi-step and multi-stage time integration schemes. These schemes advance the solution from time node n to $n+1$, $\mathbf{U}^n \rightarrow \mathbf{U}^{n+1}$, over a time step of size Δt, using state information at n and possibly previous time nodes. This subsection summarizes the formulas for the methods used.

BDF An n_{step} backwards differentiation (BDF) formula takes the form

$$\frac{\mathbf{M}}{\Delta t}\sum_{i=0}^{n_{\text{step}}} a_i \mathbf{U}^{n+1-i} + \mathbf{R}(\mathbf{U}^{n+1}, t^{n+1}) = \mathbf{0}, \tag{4}$$

where for BDF1, $n_{\text{step}} = 1$, $a_i = [1, -1]$, and for BDF2, $n_{\text{step}} = 2$, $a_i = [\frac{3}{2}, -2, \frac{1}{2}]$. BDF2 requires a different startup scheme, here taken as BDF1. In addition, BDF3 and beyond are no longer A-stable, and hence not used in this work.

(S)DIRK An n_{stage} diagonally-implicit Runge-Kutta (DIRK) method for advancing the state from $\mathbf{U}^n$ to $\mathbf{U}^{n+1}$ over a time step of size Δt takes the form

$$\begin{aligned}
&\text{for} \quad i = 1 : n_{\text{stage}} \\
&\qquad \mathbf{S}_i = -\frac{\mathbf{M}}{\Delta t}\mathbf{W}^0 + \sum_{j=0 \text{ or } 1}^{i-1} a_{ij}\mathbf{R}(\mathbf{W}^j, t^n + b_j\Delta t) \\
&\qquad \text{solve: } \frac{\mathbf{M}}{\Delta t}\mathbf{W}^i + a_{ii}\mathbf{R}(\mathbf{W}^i, t^n + b_i\Delta t) + \mathbf{S}_i = \mathbf{0} \\
&\text{end}
\end{aligned} \tag{5}$$

where $\mathbf{W}^0 = \mathbf{U}^n$ is the state at the start of the time interval, $\mathbf{U}^{n+1} = \mathbf{W}^{n_{\text{stage}}}$ is the desired result, and $t^n + b_i\Delta t$ are the stage times. When the on-diagonal coefficients a_{ii} are identical, these methods are called singly-diagonally-implicit Runge Kutta (SDIRK) [20], although this designation is not ubiquitous [21]. Presently, we will omit the S from SDIRK for consistency with previous published work. When the summation in the construction of $\mathbf{S}$ begins at $j = 0$, the methods are called (first-stage) explicit DIRK methods, to denote that there is no system solution, only a residual evaluation, at "stage 0". The designation ESDIRK then means that the summation starts at $j = 0$ and that the a_{ii} are identical.

The coefficients a_{ij} and b_i define the method. A third-order accurate, three-stage scheme used in this work, DIRK3, has coefficients [21]

$$a_{ij} = \begin{bmatrix} \alpha & 0 & 0 \\ \tau - \alpha & \alpha & 0 \\ \beta_1 & \beta_2 & \alpha \end{bmatrix}, \quad b_i = \begin{bmatrix} \alpha \\ \tau \\ 1 \end{bmatrix}. \tag{6}$$

where α is the root of $x^3 - 3x^2 + \frac{3}{2}x - \frac{1}{6} = 0$ lying in $(\frac{1}{6}, \frac{1}{2})$, $\tau = (1 + \alpha)/2$, $\beta_1 = -(6\alpha^2 - 16\alpha + 1)/4$, and $\beta_2 = (6\alpha^2 - 20\alpha + 5)/4$. A fourth-order accurate DIRK4 scheme with $n_{\text{stage}} = 5$ is [22]

$$a_{ij} = \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 \\ \frac{17}{50} & -\frac{1}{25} & \frac{1}{4} & 0 & 0 \\ \frac{371}{1360} & -\frac{137}{2720} & \frac{15}{544} & \frac{1}{4} & 0 \\ \frac{25}{24} & -\frac{49}{48} & \frac{125}{16} & -\frac{85}{12} & \frac{1}{4} \end{bmatrix}, \quad b_i = \begin{bmatrix} \frac{1}{4} \\ \frac{3}{4} \\ \frac{11}{20} \\ \frac{1}{2} \\ 1 \end{bmatrix}. \tag{7}$$

For the ESDIRK methods, the indices of the coefficients a_{ij} and b_i begin at 0. Appendix A presents these for the fourth and fifth-order accurate schemes, ESDIRK4 and ESDIRK5, used in this work.

MEBDF Modified extended backwards differentiation formulae (MEBDF) [23, 24] build on the standard backwards differentiation formulae, using additional system solutions, to improve stability at high-orders of accuracy. Denoting by BDF($n \to n+1$) the n_{step} BDF formula that advances the state from time node n to $n+1$, the corresponding MEBDF scheme consists of the following steps:

$$\begin{aligned}
&\bar{\mathbf{U}}^{n+1} = \text{BDF}(n \to n+1) \\
&\bar{\mathbf{U}}^{n+2} = \text{BDF}(n+1 \to n+2) \quad \text{using } \bar{\mathbf{U}}^{n+1} \text{ at } n+1 \\
&\text{solve for } \mathbf{U}^{n+1}\text{: } \frac{\mathbf{M}}{\Delta t}\mathbf{U}^{n+1} + (\beta_0 - v_0)\mathbf{R}(\mathbf{U}^{n+1}) + \sum_{j=0}^{1-n_{\text{step}}} a_j \mathbf{U}^{n+j} + v_0\mathbf{R}(\bar{\mathbf{U}}^{n+1}) + \beta_1\mathbf{R}(\bar{\mathbf{U}}^{n+2}) = \mathbf{0}
\end{aligned} \tag{8}$$

The MEBDF schemes are A-stable and require additional coefficients: the n_{step} values a_j, β_0, β_1. A nonzero value of v_0 distinguishes MEBDF from the original EBDF schemes [25], and it is given by $v_0 = \beta_0 - 1/a_0$. MEBDF3 is a three-step, fourth-order accurate method with coefficients

$$a_j = \left[-\frac{279}{197}, \frac{99}{197}, -\frac{17}{197}\right], \quad \beta_0 = \frac{150}{197}, \quad \beta_1 = -\frac{18}{197}. \tag{9}$$

Note that MEBDF requires uniform time steps and another startup scheme for the first $n_{\text{step}} - 1$ steps, here taken as DIRK of one order lower than the MEBDF order (e.g. DIRK3 for MEBDF3).

SAMF The split Adams-Moulton family (SAMF) [26] of schemes consist of two stages, the first of which is a predictor,

$$\frac{\mathbf{M}}{\Delta t}\frac{1}{b_0 + c_0\theta}\bar{\mathbf{U}} + \mathbf{R}(\bar{\mathbf{U}}, t^{n+1}) + \mathbf{S}_0 = \mathbf{0}, \tag{10}$$

where

$$\mathbf{S}_0 = \frac{\mathbf{M}}{\Delta t}\left(-(1 + a_{\text{sum}}\theta)\mathbf{U}^n + \sum_{i=2}^{n_{\text{step}}} a_i\theta\mathbf{U}^{n+1-i}\right) + \sum_{i=1}^{n_{\text{step}}}(b_i + c_i\theta)\mathbf{R}(\mathbf{U}^{n+1-i}, t^{n+1-i}), \tag{11}$$

and $a_{\text{sum}} = \sum_{i=2}^{n_{\text{step}}} a_i$. The second stage is a corrector,

$$\frac{\mathbf{M}}{\Delta t}\mathbf{U}^{n+1} + (b_0 + c_0\theta)\mathbf{R}(\mathbf{U}^{n+1}, t^{n+1}) + \mathbf{S}_1 = \mathbf{0}, \tag{12}$$

where

$$\mathbf{S}_1 = -\frac{\mathbf{M}}{\Delta t}\mathbf{U}^n + \sum_{i=1}^{n_{\text{step}}} b_i\mathbf{R}(\mathbf{U}^{n+1-i}, t^{n+1-i}) + \theta\mathbf{R}(\bar{\mathbf{U}}, t^{n+1}). \tag{13}$$

Specifically, SAMF3 is a three-step, two-stage, fourth order, method with $A(89.999^\circ)$ stability (almost A-stable), with coefficients (starting at index 0),

$$\begin{aligned} \mathbf{a} &= [1, 0, -72/15, 16/15], \\ \mathbf{b} &= [9/24, 19/24, -5/24, 1/24], \\ \mathbf{c} &= [-1, 209/45, -11/9, 11/45], \\ \theta &= -15/88 - 3\sqrt{5}/22. \end{aligned} \tag{14}$$

For the initial time steps of this multi-step scheme, we use DIRK3.

III. Time-Step Control

We develop a time-step control strategy that balances the temporal and spatial errors in an unsteady simulation. The temporal error can be defined as the difference between the solution obtained with the current temporal discretization and one with an infinitely-refined temporal discretization, e.g. using an infinitesimally-small time step. The spatial error can analogously be defined as the difference between the solution with the current spatial discretization and one that is infinitely-refined, e.g. using a mesh size that approaches zero. Without access to these additional solutions, we need alternative ways to measure the errors. One possibility is an output error, through an adjoint-weighted residual. However, computing an unsteady adjoint is expensive, due to the requisite backwards-in-time integration on a finer space. An adjoint approach would also require the selection of one scalar output. An alternative, used in this work, is an unweighted residual, which is straightforward to measure and requires no extra solutions. Although unweighted residuals by themselves do not always yield accurate spatial adaptive indicators, particularly for advection-dominated flows, the relative effect of spatial and temporal discretization on the residual is still informative in choosing the time step.

We consider the error made in one time step, from t^n to $t^{n+1} = t^n + \Delta t$. Even if the state $\mathbf{U}^n$ exactly approximates the true solution at t^n, the computed solution at $\mathbf{U}^{n+1}$ will be polluted by errors from two sources. First, the exact

solution for $t > t^n$ will generally not lie in the finite-dimensional spatial approximation space, i.e. the span of the finite-element basis functions in our case. The resulting *spatial error* arises from an incorrect representation of the state and its time rate of change, the velocity. Second, the time integration scheme will generally not exactly capture the temporal evolution of $\mathbf{U}(t)$, except in certain special cases such as when $\mathbf{U}$ is constant or varies as a low-order polynomial in time. The resulting *temporal error* depends on the time integration scheme and step size.

As defined, both the spatial and temporal errors are local, meaning that they are made in one time step. Reducing the spatial error requires a finer mesh or a higher spatial order. Reducing the temporal error requires a smaller time step or a scheme with a higher temporal order of accuracy. The errors are also generally coupled: for example, changing the spatial approximation space affects the representation of the state and velocity, which then changes the state trajectory through the time step and affects the temporal error.

Our goal is to control the time step, Δt, to make the temporal error comparable to the spatial error. This task requires a definition of error that is not overly expensive to compute and that allows for a direct comparison of spatial and temporal errors. We use a residual-based approach and note that, by Eqn. 3, the spatial residual is a scaled state velocity,

$$\mathbf{R}(\mathbf{U}) = -\mathbf{M}\frac{d\mathbf{U}}{dt}. \tag{15}$$

Errors in the spatial residual thus correspond to errors in the velocity. Both the spatial and the temporal discretizations affect the velocity, and hence the velocity error provides a consistent means of comparing discretizations.

A. Spatial Error

We first consider the spatial error. Measuring the spatial error requires a comparison of the solution relative to one on a finer approximation space. In the present work, we use an order-incremented finer space, $p \longrightarrow p+1$, which has more basis functions and a richer approximation. We denote by the subscripts H and h the current and fine approximation spaces, respectively. Let $\mathbf{U}_h^H = \mathbf{I}_h^H \mathbf{U}_H$ be the prolongation of the coarse state to the fine space. The prolongation operator $\mathbf{I}_h^H$ is lossless for the $p+1$ fine space, and it can act on either the state or the velocity.

Evaluating the fine-space residual with the injected state yields a certain spatial residual, $\mathbf{R}_h(\mathbf{U}_h^H)$. Unlike in steady problems, we do not expect $\mathbf{R}_h$ to be close to zero, as the state will generally have a nonzero velocity. To measure the spatial error, we therefore compare the velocity on the coarse space with the one on the fine space. This comparison must be done on the same approximation space, taken as h, and hence the coarse-space velocity is prolongated to the fine space. The resulting spatial error, computed at a particular time node (here at the end of the time step, $n+1$), is

$$\varepsilon_{\text{space}} \equiv \left\| \mathbf{R}_h(\mathbf{I}_h^H \mathbf{U}_H^{n+1}) - \mathbf{M}_h \mathbf{I}_h^H \mathbf{M}_H^{-1} \mathbf{R}_H(\mathbf{U}_H^{n+1}) \right\|. \tag{16}$$

The residual error norm can be a standard one such as L_2, or a more careful averaging, as discussed in Section III.C. Multiplying the terms inside by the inverse fine-space mass matrix yields a velocity error, but we have observed improved performance with residuals, particularly on adapted meshes. For problems with singularities, such as at airfoil trailing edges, large velocity errors can be observed at the location of the singularity when incrementing the spatial order, even when the elements there are small. Using residuals instead of velocities prevents these, usually already adapted, elements from dominating the spatial error.

B. Temporal Error

Next, we consider the temporal error. Some time-marching methods have built-in error estimates that provide differences in states between low and high-order time integration. Converted into velocities, such errors could be used to define the temporal error. However, not all time-marching methods come equipped with such error estimates. Therefore, we use a simplified projection-based error estimate obtained by reconstructing the state as an appropriate-order polynomial in time, $\mathbf{U}(t)$, and projecting it to a lower order polynomial in time, $\tilde{\mathbf{U}}(t)$. The temporal reconstruction is obtained by using endpoint states and velocities [5], and for the schemes considered in this work, only cubic reconstruction is employed. The projection to a lower temporal order, here quadratic, is L_2 over the time step $n \longrightarrow n+1$, with the constraint that the state at n does not change. The temporal error is then the difference between the velocities, measured as fine-space residuals, at time node $n+1$ using the original state and the state from the lower-order temporal reconstruction:

$$\varepsilon_{\text{time}} \equiv \left\| \mathbf{R}_h(\mathbf{I}_h^H \mathbf{U}_H^{n+1}) - \mathbf{R}_h(\mathbf{I}_h^H \tilde{\mathbf{U}}_H^{n+1}) \right\|. \tag{17}$$

Figure 1 summarizes the procedure for measuring the spatial and temporal errors, using a color coding consistent with that given in Eqns. 16 and 17.

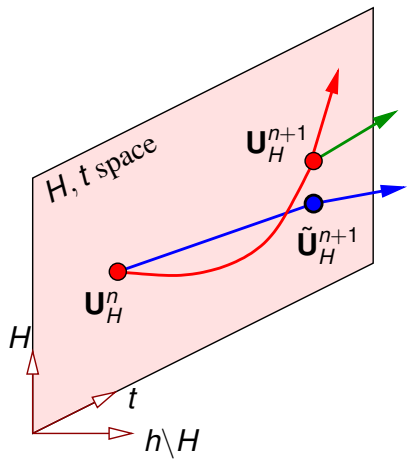
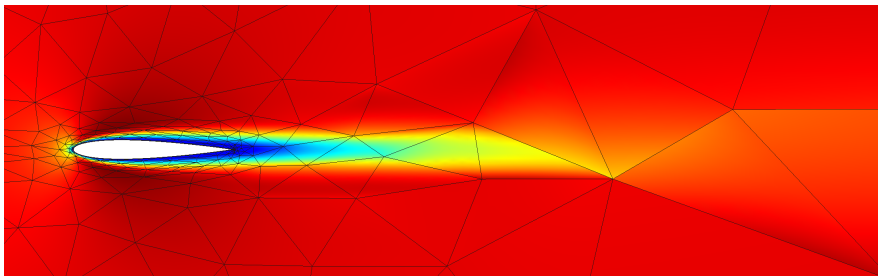


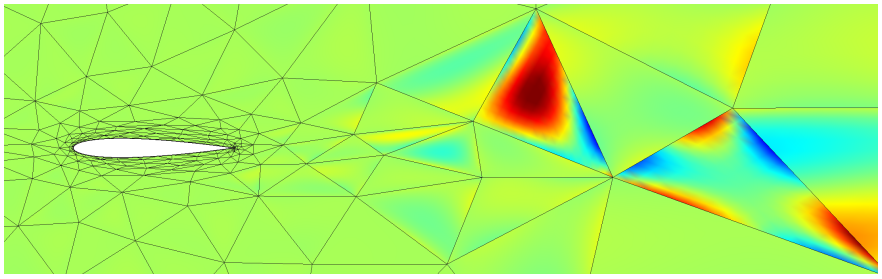
Fig. 1 Spatial and temporal error estimates via velocities at the end of one time step, $n \rightarrow n+1$. The colored arrows are velocities that correspond to the residuals in Eqns. 16 and 17.

C. Element-based Averaging

Both Eqn. 16 and Eqn. 17 involve norms of residuals. A standard discrete norm, such as L_1 or L_2, computed over all degrees of freedom can be sensitive to outliers and hence can perform poorly. We have observed this on output-adapted meshes, where certain areas of the domain that have little impact on the output are not well resolved, for example as shown in Figure 2, leading to large spatial residuals on the elements in these regions. These spatial residuals then



(a) State



(b) Residual

Fig. 2 Large spatial residuals in intentionally under-resolved regions of adapted meshes, here in the viscous wake of an airfoil, can dominate residual-based spatial error estimates.

dominate the error estimates, and the adaptive time stepping algorithm suffers. Specifically, the large overall spatial error makes the time step too large when the errors are balanced, adversely affecting the temporal accuracy in the resolved regions. A proper means of resolving this problem would require adjoints as residual weights, but unsteady adjoints would add a tremendous computational cost. Fortunately, we have found a simpler approach that works well on such adapted meshes. This approach relies on the observation that these meshes, by virtue of being adapted, have large

elements in the under-resolved regions, so that the large spatial residuals occur over a small number of elements. By using an element-based averaging approach, outlier effects can be mitigated.

Denote by the subscript e norms (here L_2) computed over a single element. So element e has a spatial error of $\varepsilon_{\text{time},e}$ and a temporal error of $\varepsilon_{\text{time,e}}$. Define the temporal error fraction on element e as

$$f_{\text{time},e} \equiv \frac{\varepsilon_{\text{time,e}}}{\varepsilon_{\text{time,e}} + \varepsilon_{\text{space,e}}}. \tag{18}$$

The total temporal error fraction over the entire mesh is then computed by averaging $f_{\text{time},e}$ over elements,

$$f_{\text{time}} = \frac{1}{N_e} \sum_e f_{\text{time},e}. \tag{19}$$

D. Control Algorithm

The spatial and temporal error estimates are calculated after taking a time step. Let Δt be the size of this step. The time-step control algorithm must then decide whether to redo this time step, if the temporal error is too high compared to the spatial error, or whether to proceed to the next time step. If the latter, the algorithm should pick the size of the next time step to balance the errors. This choice is done using the current-step error estimates, and hence the decision about the size of the next step will be retrospective. In this work, we set the time step using a relatively simple strategy based on an assumed temporal order, r. The strategy proceeds as follows:

1) Take the current time step, estimate the errors, and compute f_{time} according to Eqn. 19.
2) If $f_{\text{time}} > f_{\text{limit}}$, redo the time step using $\Delta t/2$.
3) Else, set $\Delta t_{\text{next}} = \min(r_{\max}, f^{-1/r})\Delta t$, where $f \equiv f_{\text{time}}/(1 - f_{\text{time}})$ is the temporal/spatial error ratio.

For the present results, we use $f_{\text{limit}} = 0.6,\ r_{\max} = 1.5,\ r = 2$, which may not be optimal but have yielded reasonable results. Better choices for these parameters, basing r on the formal temporal order of accuracy, and a more sophisticated control strategy are the subject of future work.

Finally, we note that much previous work exists in adaptive time stepping for ordinary differential equations [20, 27]. The novelty of the present approach is the idea of balancing temporal and spatial errors, which only applies when the system of ODEs arises from partial differential equations. Rather than keeping the temporal error constant, we peg it to the spatial error to expend the appropriate amount of computational effort on time accuracy for a given spatial discretization. Recent work exists with the same intent [28], using an embedded time-stepping scheme. Our work differs in its support for general time-stepping schemes, and in its use of a simple element-based average norm suitable for output-based adapted meshes.

IV. Results

In this section, we present results that compare the efficiency of the various time-marching schemes, and the performance of the time-step control strategy. Three problems serve as the testing platform: a manufactured solution for the scalar advection-diffusion equation, and two Navier-Stokes simulations on deforming domains.

A. Manufactured Solution

We first consider a manufactured solution for the two-dimensional scalar advection-diffusion equation, which takes the following simplified form of Eqn. 1,

$$\frac{\partial u}{\partial t} + \nabla \cdot \left(\vec{V} u - \mu \nabla u \right) + S(u) = 0. \tag{20}$$

The source term $S(u)$ drives $u(x, y, t)$ to the following manufactured solution,

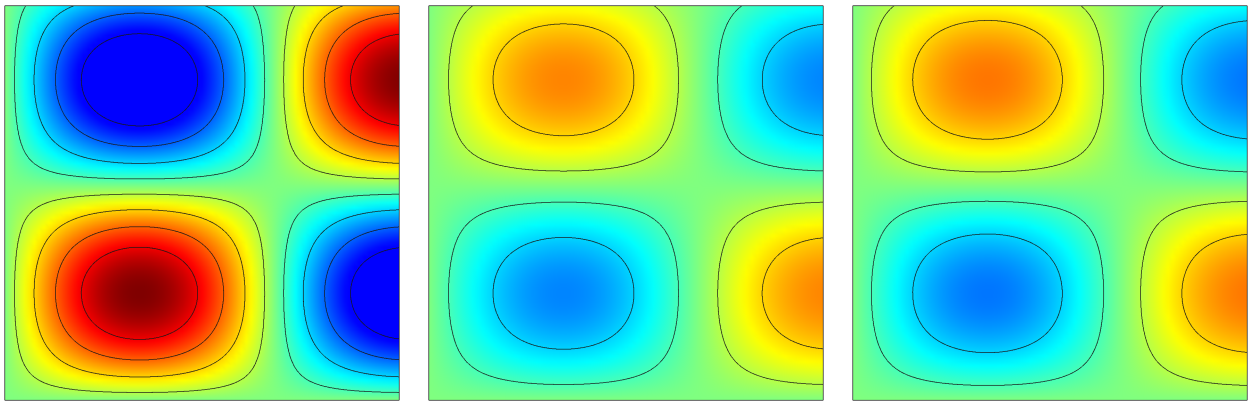
$$u^{\text{MS}}(x, y, t) = \sin(ax) \sin(by) \cos(ct). \tag{21}$$

Presently, we choose a square 2×2 domain, Ω, with the lower-left corner as the origin, $\vec{V} = [0.8, 0.6]$, $\mu = 0.5$, and $(a, b, c) = (2.3, 2.9, 4.2)$. The unsteady manufactured solution is prescribed as a full-state boundary condition on all

four edges of the square. The time horizon is from $t = 0$ to $t = T = 2$, and the output of interest is the L_2 norm of the state error at the final time,

$$L_2 \text{ state error} = \left(\frac{1}{|\Omega|} \int_\Omega \left(u(x, y, T) - u^{\text{MS}}(x, y, T) \right)^2 \, d\Omega \right)^{\frac{1}{2}}. \tag{22}$$

At time $t = 0$ the state is initialized using least-squares projection of u^{MS}. Figure 3 shows the state at various times in the simulation.



(a) $t = 0$ (b) $t = 1$ (c) $t = 2$

Fig. 3 Scalar manufactured solution states at various times.

The test matrix for the convergence study consists of five time schemes: BDF2, DIRK3, MEBDF3, SAMF3, and ESDIRK4. For each scheme, simulations are performed on $p = 2, 3$, three grid sizes (4×4, 8×8, 16×16, split into triangles), and eight numbers of time intervals: $N_t = 2, 4, 8, 16, 32, 64, 128, 256$, for a total of 48 simulations per scheme. Figure 4 shows the final-time L_2 error plotted versus time step size Δt for all of the $p = 2$ runs. Lines connect data for a single mesh and time scheme, with N_t varying.

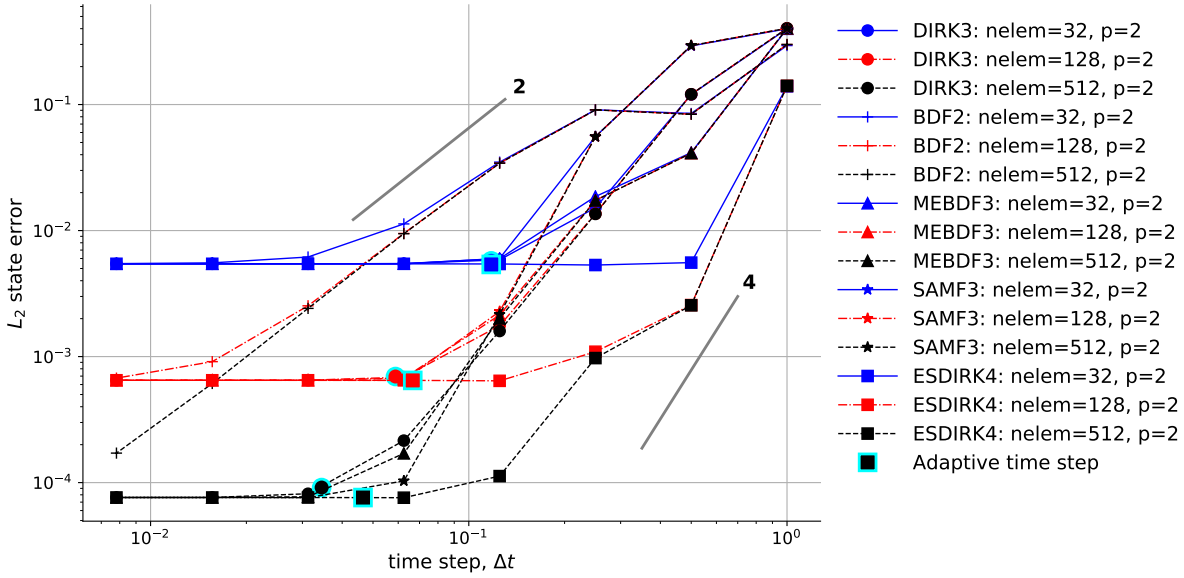


Fig. 4 Manufactured solution output convergence results, $p = 2$.

We observe the expected behavior: with increasing N_t, the L_2 error first decreases as a large temporal error is

reduced, but then stalls to a constant value of the spatial error when the latter begins to dominate. The finest spatial resolution run of BDF2 does not reach this plateau, as its relatively low temporal order keeps the temporal error dominant for the N_t values simulated. Not all of the time schemes reach their asymptotic theoretical convergence rates prior to the spatial error taking over, but the higher-order schemes do show more rapid temporal convergence. In addition, we note that the order of accuracy of each scheme was also verified separately on a fixed spatial discretization, using a truth solution with a large number of time steps.

Figure 5 shows the error convergence results for $p = 3$. Compared to $p = 2$, the errors now stall out at lower values, as expected from the higher spatial approximation order. As a result, BDF2 only bottoms out for the coarsest mesh. Otherwise, the start of the curves, at small N_t and large Δt, is the same as for $p = 2$, since this is the region where temporal error dominates.

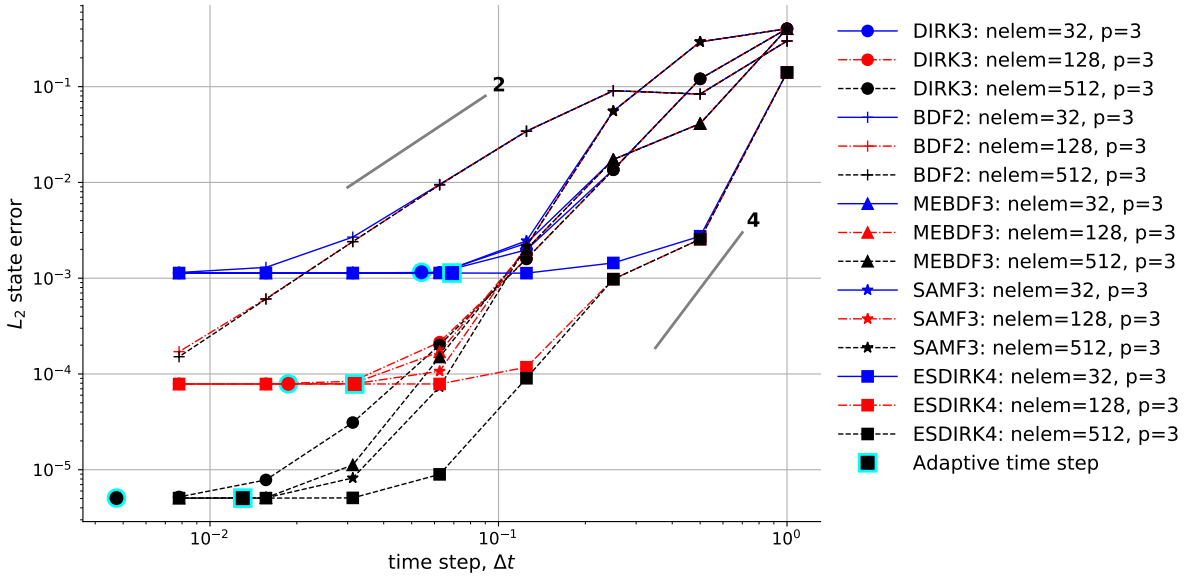


Fig. 5 Manufactured solution output convergence results, $p = 3$.

In both Figures 4 and 5, the larger highlighted markers correspond to the adaptive time-step results. These are obtained only for the DIRK3 and the ESDIRK4 schemes, which do not require uniform time stepping. For each of these runs, N_t is initially set to a coarse value of 4. The time step control strategy then sets the time step using comparisons to the spatial error, as described in Section III. This leads to a sharp reduction in the initial time step, which is redone several times as the temporal error is found dominant. The time step then varies through the simulation to maintain balance with the spatial error. The location of the resulting data point on the plot is determined by using the actual error and the average time step, T/N_t.

Ideally, we expect the results of the adaptive time-step runs to lie at the "kink" of the corresponding uniform time-step refinement results, where the errors begin to stall out – perhaps further to the right when the problem calls for non-uniform time steps. For this temporally smooth and periodic problem, however, the optimal time step distribution is close to uniform. And indeed, the time-step control results do lie close to the kinks of the uniform refinement results. The important point is that the control strategy is able to determine the time step automatically, even when starting from a coarse temporal discretization, eliminating the need for a temporal convergence study with multiple simulations.

Finally, back to uniform time stepping, we are also interested in the efficiency of the various time-marching schemes. We can obtain this information from the data in Figures 4 and 5, by plotting the errors versus the number of nonlinear solutions required, instead of versus Δt. The number of nonlinear solutions per time step ranges from 1 for BDF2, to 5 for ESDIRK4. Figure 6 shows the resulting data. We see that the spread in the lines is now much smaller, since as expected, the higher-order time-marching schemes require more stages. At low error levels, BDF2 performs on par with the other methods, but this is still where temporal error dominates for all of the spatial discretizations. At more strict error tolerances, the SAMF3 method becomes most efficient, due to its low cost of two stages per time step, followed by ESDIRK4, due to its high accuracy compensating for its high cost of five stages per time step.

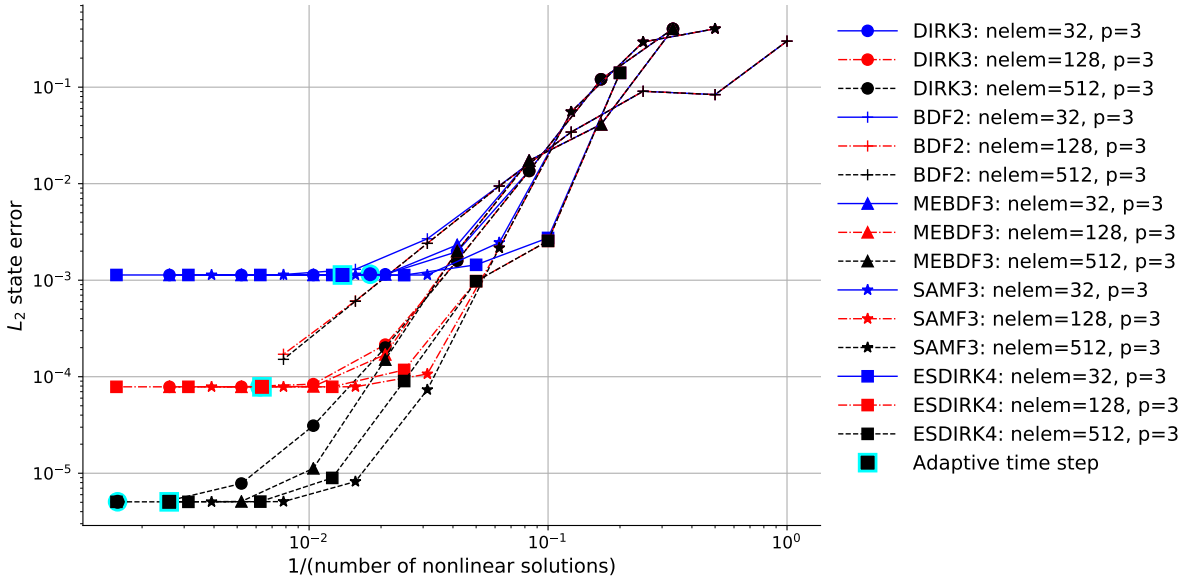


Fig. 6 Manufactured solution output convergence efficiency results, versus total number of nonlinear solutions, for $p = 3$.

B. Deforming Cylinder

For the second test case, we consider the solution of the Navier-Stokes equations inside a two-dimensional cylinder undergoing translation, rotation, and deformation. The starting point for this case is one of the High-Fidelity CFD workshop verification cases [29]. The unit-diameter cylinder initially centered at the origin undergoes three simultaneous motions: elliptical deformation, rotation, and translation. The net mapping from reference coordinates, (X, Y), to global coordinates, (x, y), can be written as a combination of these motions,

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & p_x(t) \\ 0 & 1 & p_y(t) \\ 0 & 0 & 1 \end{bmatrix}}_{\text{translation}} \underbrace{\begin{bmatrix} \cos(\theta(t)) & -\sin(\theta(t)) & 0 \\ \sin(\theta(t)) & \cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \psi(t) & 0 & 0 \\ 0 & \frac{1}{\psi(t)} & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{deformation}} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \tag{23}$$

where

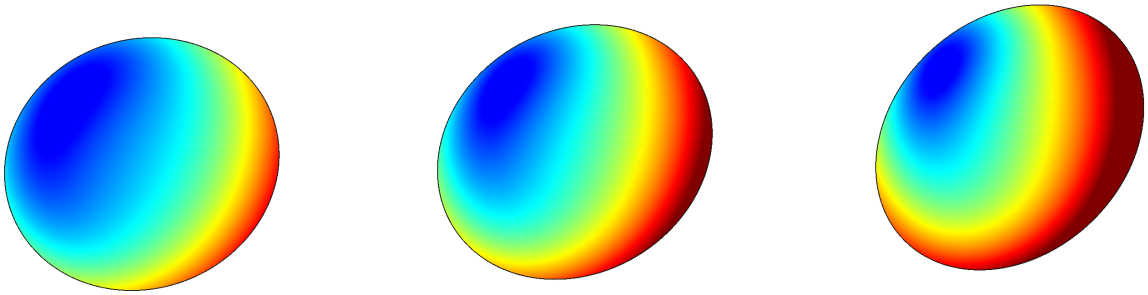
$$\psi(t) = 1 + \alpha(t), \quad \theta(t) = \pi\alpha(t), \quad [p_x(t), p_y(t)] = [0.6, 0.8]\alpha(t), \quad \alpha(t) = \begin{cases} t^3/2 - 3t^4/16 \\ 3t^2/8 - t^3/8 \end{cases}. \tag{24}$$

The two expressions for $\alpha(t)$ correspond to startup at $t = 0$ of different smoothness: t^3 for the first one, and t^2 for the second one. The mesh motion is implemented through an arbitrary Lagrangian-Eulerian approach with an analytical mapping [14, 30]. The initial condition is stagnant fluid, with state vector

$$\mathbf{u} = [\rho, \rho u, \rho v, \rho E] = [1, 0, 0, 10]. \tag{25}$$

Note that the state vector is given in dimensional but convenient $O(1)$ units. The dynamic viscosity is set to a constant value of $\mu = 0.1$, the specific heat ratio is $\gamma = 1.4$, and the Prandtl number is 0.72. The simulation time horizon is from $t = 0$ to $t = T = 0.9$. Figure 7 shows the solution at several times in the simulation.

The test matrix for the convergence study consists of five time schemes: BDF2, DIRK3, MEBDF3, SAMF3, and ESDIRK5. For each scheme, simulations are performed on $p = 3$, three grid sizes (20, 80, 320 quadrilaterals), and seven numbers of time intervals: $N_t = 4, 8, 16, 32, 64, 128, 256$, for a total of 21 simulations per scheme. Two outputs are of interest in this case: (1) the final lift, defined as the vertical component of the force exerted by the fluid on the cylinder at $t = T$, and (2) the vertical impulse, defined as the integral of the lift over the time horizon. Figure 8 shows



(a) $t = 0.675$ (b) $t = 0.7875$ (c) $t = 0.9$

Fig. 7 Cylinder geometry and Mach number contours at various times.

the final lift error plotted versus time step size Δt for the $p = 2$ runs and t^2 startup. Lines connect data for a single mesh and time scheme, with N_t varying.

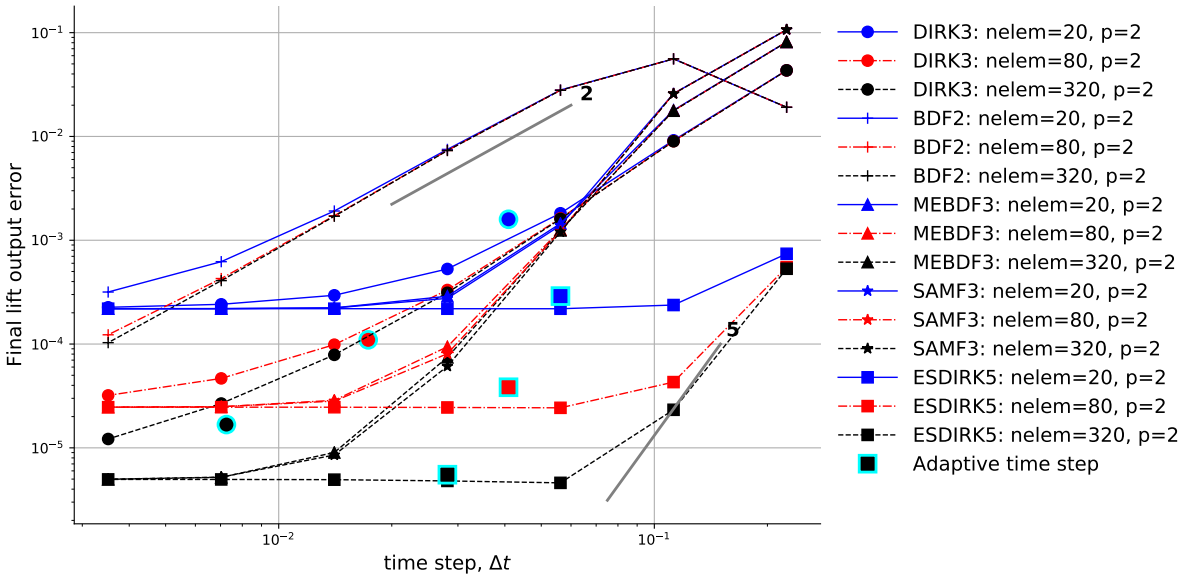


Fig. 8 Cylinder final lift convergence results, $p = 2$ and t^2 startup.

We observe the expected behavior: with increasing N_t, the output error first decreases, at the appropriate convergence rate, as a large temporal error is reduced, but then stalls to a constant value of the spatial error when the latter begins to dominate. In this case, BDF2 does not reach this plateau for the N_t values tested.

The highlighted markers plotted in Figure 8 are the adaptive time step results for DIRK3 and ESDIRK5. For each of these runs, N_t is initially set to a coarse value of 8. The time step control strategy reduces the initial time step, which is redone several times and varies through the simulation to maintain balance with the spatial error, and the location of the resulting data point on the plot is determined by using the average time step, T/N_t. The time-step control strategy performs reasonably well in identifying the appropriate balance between spatial and temporal errors for both DIRK3 and ESDIRK5.

Next, we consider the impulse output. Figure 9 shows the convergence of the error in the impulse for $p = 2$ and the t^2 startup. We see that for this output, which involves an integral over the entire time horizon, all of the schemes are limited in their temporal convergence rate to 2, when measured through a uniform time-step refinement study. The temporal error dominates, and the flattening out of the error is not observed for the N_t values tested. On the other hand,

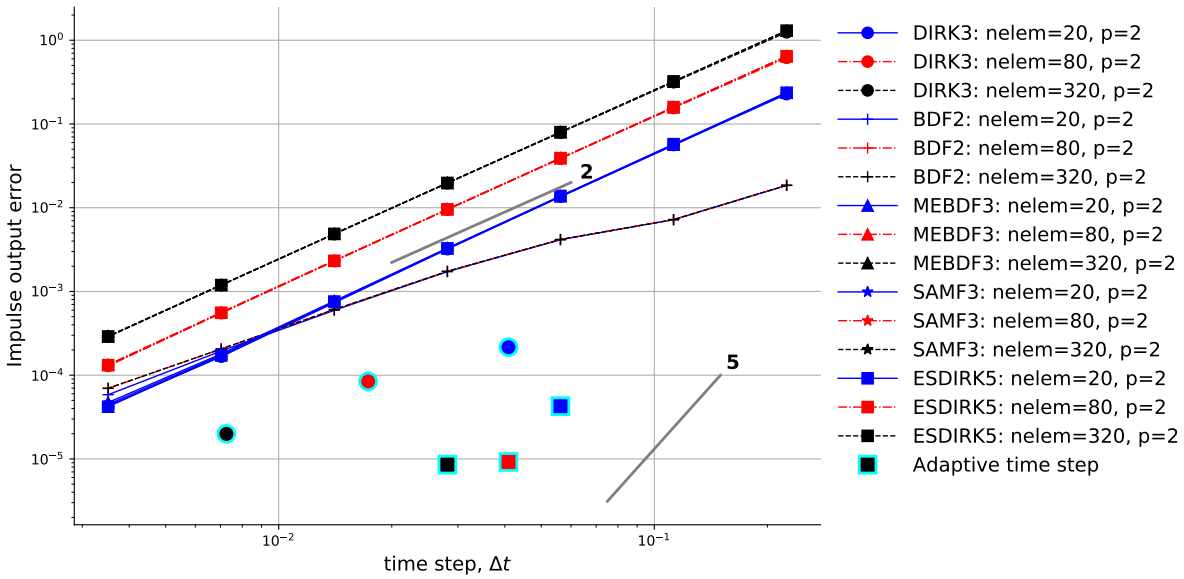


Fig. 9 Cylinder impulse convergence results, $p = 2$ and t^2 startup.

the adaptive time step results, shown by the highlighted markers, exhibit a much lower error with fewer overall time steps. This is due to the control algorithm choosing small time steps at the beginning of the simulation, effectively isolating the temporal (high-order) singularity caused by the t^2 startup.

Figure 10 shows the time-step history for this case, and the small time steps at the beginning are evident. The initial high temporal error fraction drops steeply as the initial time step is redone multiple times, and hence the large error values shown do not contribute to the overall error. Note the undershoot of the temporal error fraction, which ideally should be at 0.5, soon after startup. This undershoot is due to the control algorithm being able to grow Δt only by a limited factor at each step and to the fact that time steps in which the temporal error is small are not redone for efficiency reasons.

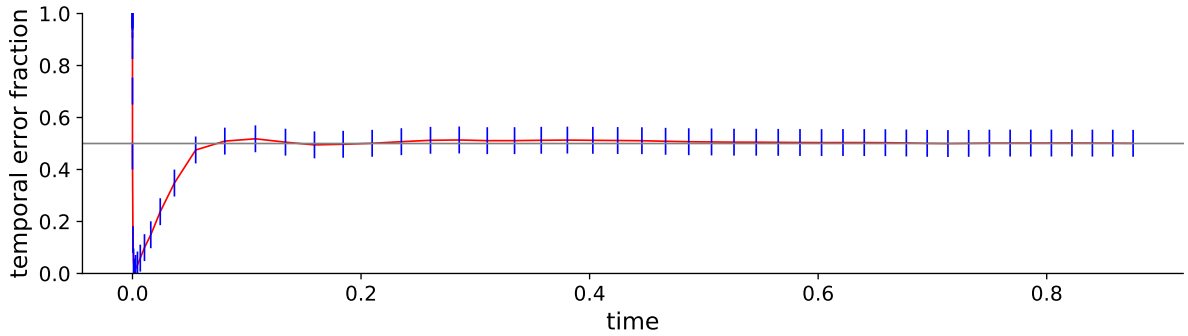


Fig. 10 Temporal error fraction and time step history for the cylinder case, $p = 2$ and t^2 startup.

Changing the startup smoothness of the problem also mitigates the temporal singularity at $t = 0$. Figure 11 shows the impulse output convergence for the smoother t^3 startup. We observe improved convergence rates from the higher-order schemes. The adaptive time step data points are also at reasonable error levels and average time step sizes, yielding balanced temporal and spatial errors.

We note that this problem has, by design, zero spatial error at $t = 0$, since the startup is from a stagnant fluid state. This causes a peculiar feature of very small initial time steps, as the temporal error is pegged to the instantaneous spatial error. At later times, the spatial error grows, and concomitantly so does the time step, making the initial small time steps

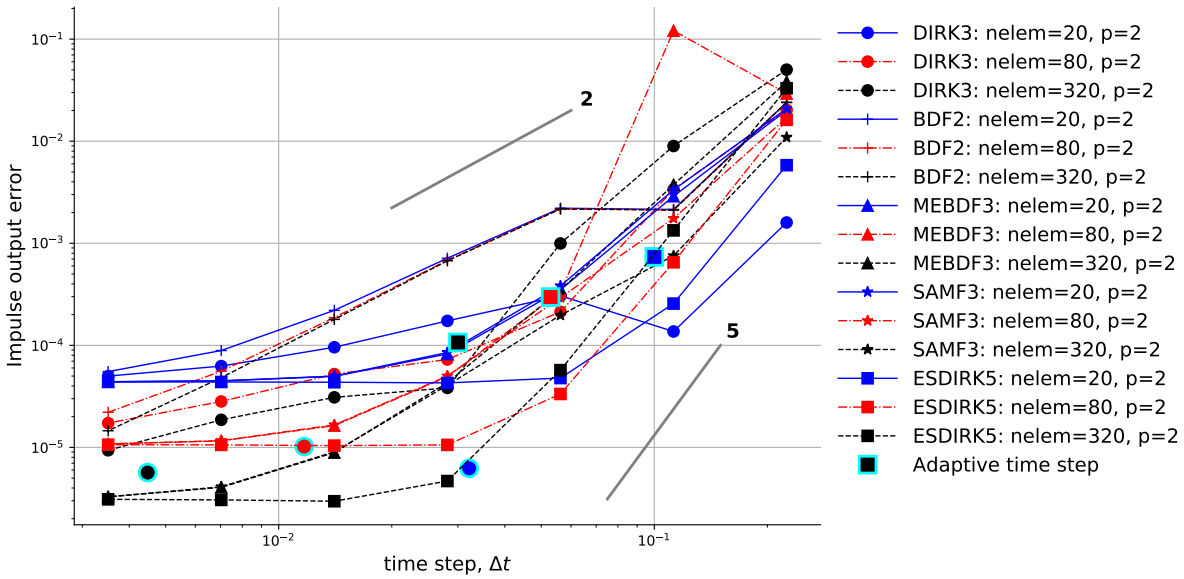


Fig. 11 Cylinder impulse convergence results, $p = 2$ and t^3 startup.

inefficient. However, without knowledge of the larger-magnitude errors to come at later times, the time-step control algorithm is inherently limited in making such a resource allocation decision.

Finally, Figure 12 shows the efficiency data for the various schemes, as the impulse error plotted versus number of nonlinear solutions required, for $p = 2$, t^3 startup. We see that the spread in the lines is smaller, as in the previous case. At strict error tolerances, the SAMF3 again method becomes most efficient, due to its low cost of two stages per time step, and its fourth-order formal temporal accuracy.

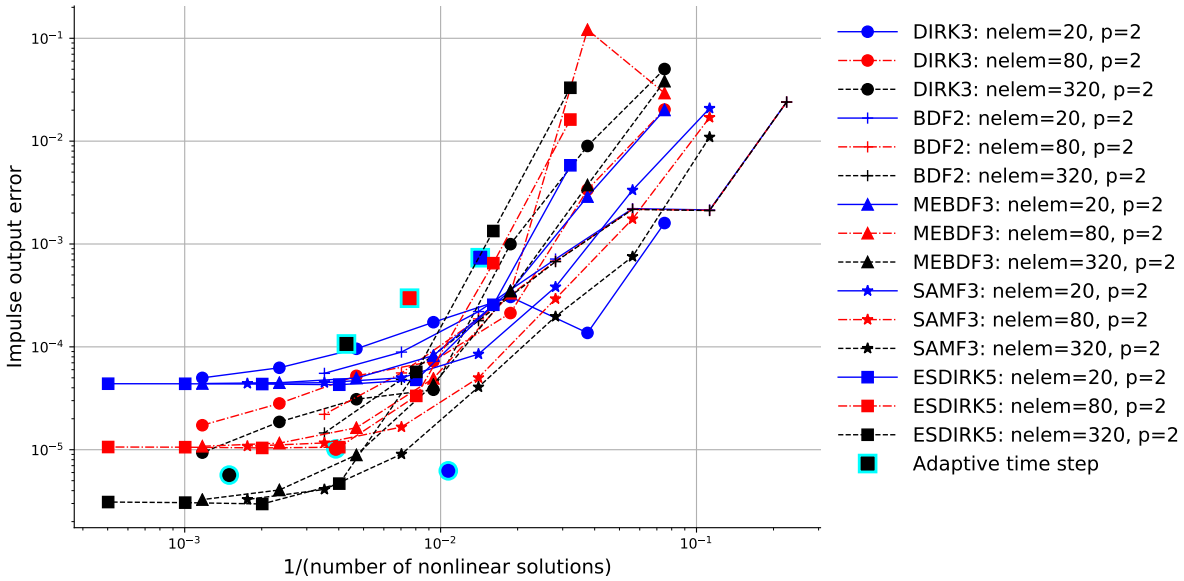


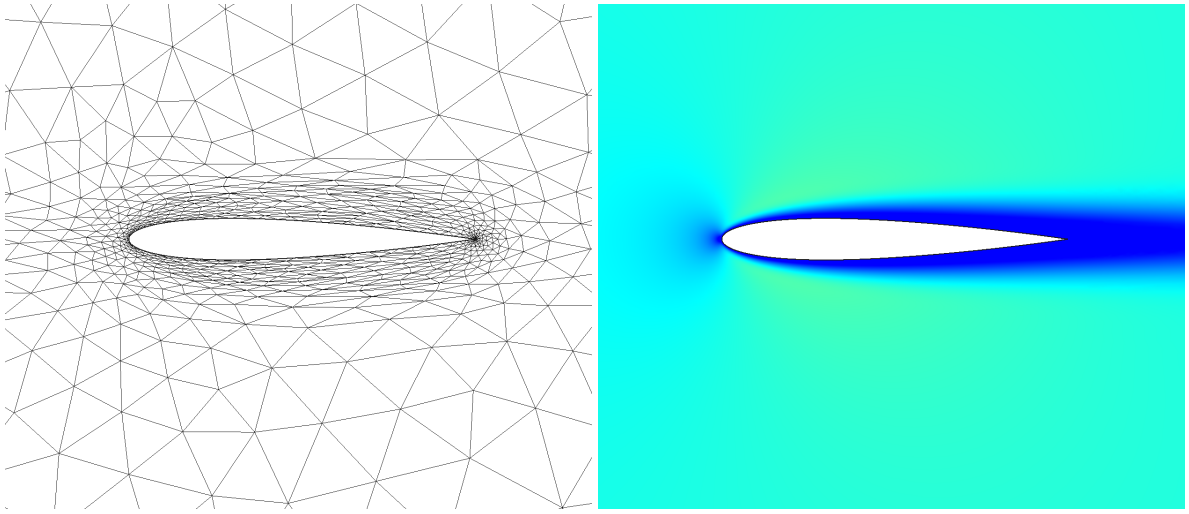
Fig. 12 Cylinder impulse convergence efficiency results, versus total number of nonlinear solutions, for $p = 2$ and t^3 startup.

C. Plunging Airfoil

For the final test case, we consider a NACA 0012 airfoil at freestream Mach number 0.1, Reynolds number 1000, and zero degrees angle of attack, undergoing plunging motion according to

$$h(t) = \frac{1}{2}t^3 - \frac{3}{16}t^4 \quad \text{or} \quad \frac{3}{8}t^2 - \frac{1}{8}t^3. \tag{26}$$

The first of these plunge motion functions is a smooth t^3 startup, whereas the second is a less-smooth t^2 startup. Figure 13 shows the finer of the two computational meshes used, which consists of cubic curved triangles on the airfoil boundary, and Mach number contours at various times.



(a) 1963-element mesh

(b) $t = 0$

(c) $t = 0.5$

(d) $t = 1.0$

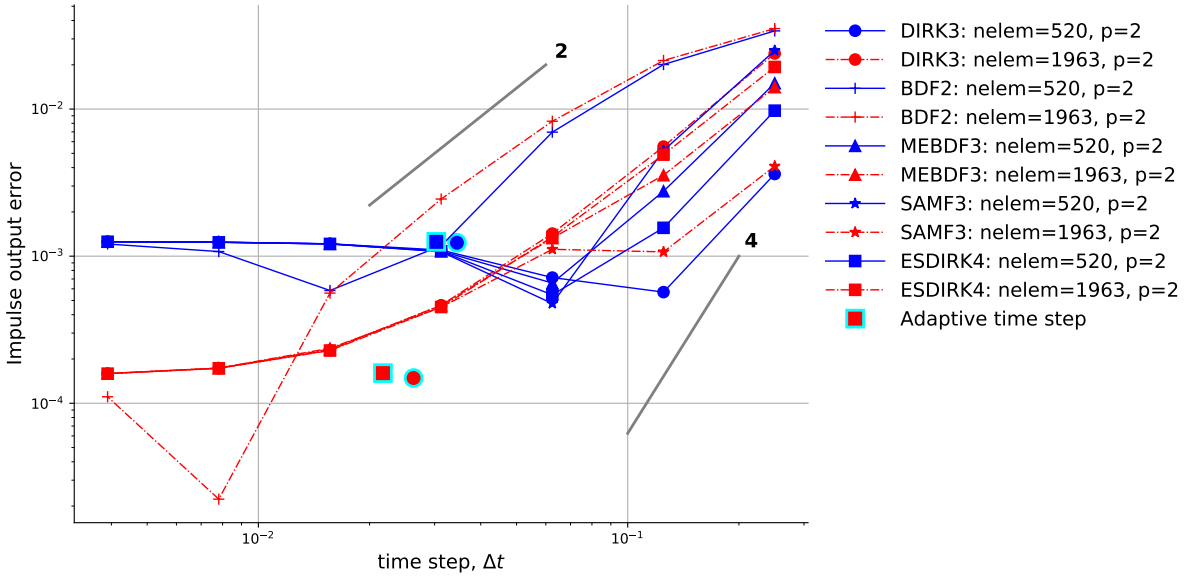
Fig. 13 Airfoil mesh and Mach number contours at various times during a plunge motion.

As in the previous case, the state vector is given in dimensional but convenient $O(1)$ units, with $\rho = 1$ and $U_\infty = 1$. The dynamic viscosity is set to a constant value of $\mu = 0.001$, the specific heat ratio is $\gamma = 1.4$, and the Prandtl number is 0.72. The simulation time horizon is from $t = 0$ to $t = T = 1.0$.

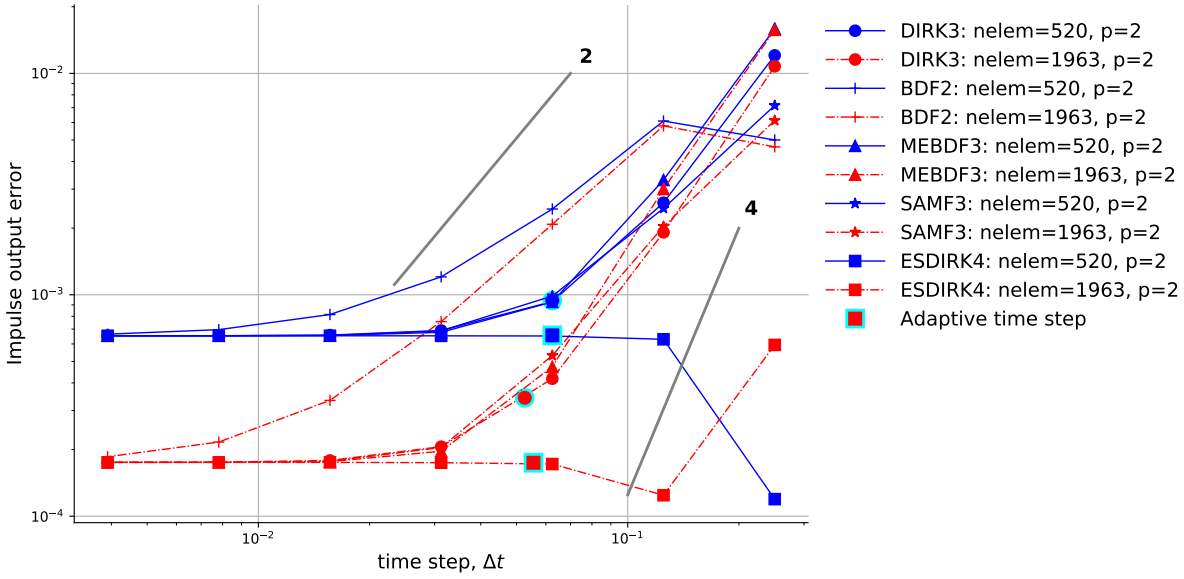
The test matrix for the convergence study consists of five time schemes: BDF2, DIRK3, MEBDF3, SAMF3, and ESDIRK4. For each scheme, simulations are performed at $p = 2$, on two grid sizes (520 and 1963 triangles), and seven

numbers of time intervals: $N_t = 4, 8, 16, 32, 64, 128, 256$, for a total of 14 simulations per scheme. The output of interest is the vertical impulse, given by the time-integral of the vertical force over the time horizon, which is $T = 1c/U_\infty$.

Figure 14 shows the impulse error plotted versus time step size Δt for the $p = 2$ runs. Lines connect data for a single mesh and time scheme, with N_t varying. We observe the expected behavior: with increasing N_t, the output error first



(a) t^2 startup



(b) t^3 startup

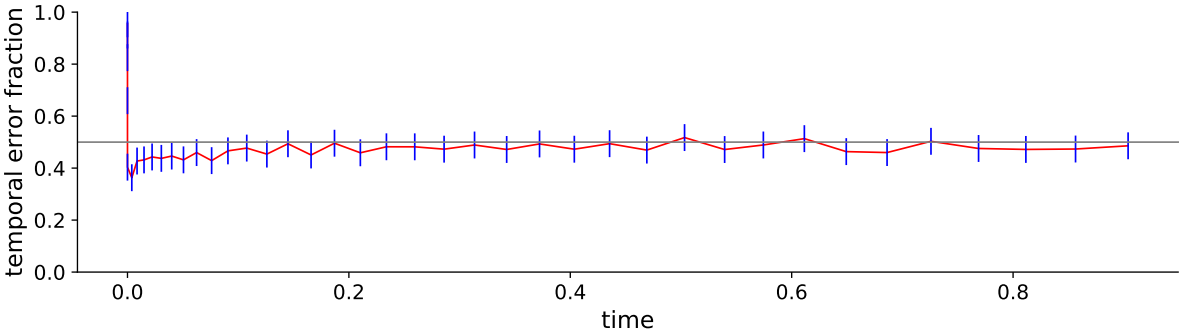
Fig. 14 Airfoil impulse convergence results, $p = 2$.

decreases, at close to the appropriate convergence rate, as a large temporal error is reduced, but then stalls to a constant value of the spatial error when the latter begins to dominate.

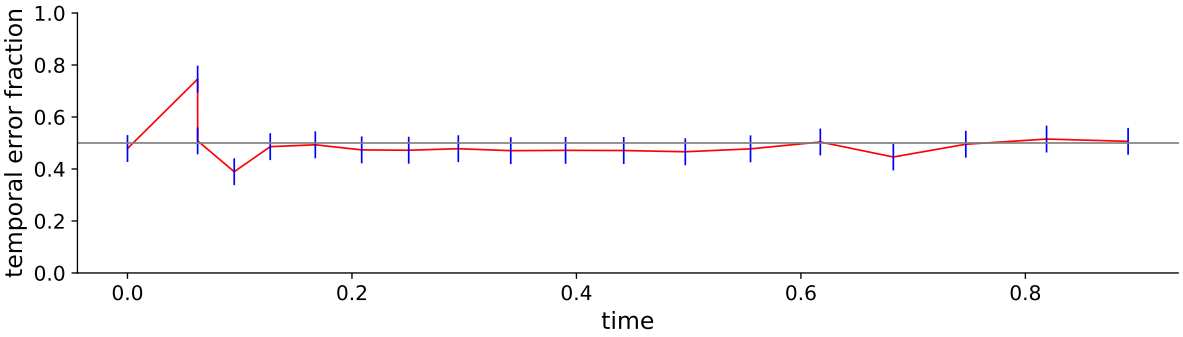
Adaptive time step results are also shown for DIRK3 and ESDIRK4. For each of these runs, N_t is initially set to a small value of 8, and the time-step control strategy reduces the initial time step, which is redone several times and varies through the simulation to maintain balance with the spatial error. The time-step control strategy performs well in

identifying the appropriate balance between spatial and temporal errors for both the t^2 and the t^3 startups.

Figure 15 shows the time-step histories for DIRK3, $p = 2$, on the 1963-element mesh. The small initial time steps for the t^2 startup are evident, set by the control strategy to reduce the high temporal error caused by a relatively-strong initial-time singularity of the motion. On the other hand, the t^3 startup yields a much more uniform time step, as the motion is smoother at $t = 0$ and the initial temporal error fraction is not as high.



(a) t^2 startup



(b) t^3 startup

Fig. 15 Temporal error fraction and time step history for the airfoil case, $p = 2$ and t^2 startup.

Finally, Figure 16 shows the efficiency data for the various schemes, as the impulse error plotted versus number of nonlinear solutions required, for $p = 3$, t^3 startup. We see that the spread in the lines is smaller, as in the previous case, but that this time ESDIRK4 is the most efficient time marching scheme, with SAMF3 coming in second. Thus, the efficiency of a particular scheme is case dependent.

V. Conclusions

This paper investigates semi-discrete time marching schemes for high-order discretizations of convection-dominated flow problems, using the discontinuous Galerkin method. It also presents a time-step control strategy that balances numerical errors arising from the temporal and spatial discretizations. While the focus is on fluid-dynamics problems, the same methods could apply to discretizations of other partial differential equations. In addition, other high-order spatial discretizations could be considered, including continuous finite elements and flux reconstruction methods. A caveat is that only implicit time marching methods have been considered: explicit methods for high-order are typically already overly resolved temporally due to stability time step restrictions.

The results for three test problems, governed by the scalar advection-diffusion and the Navier-Stokes equations, demonstrate the importance of choosing the correct time step. A time step that is too large creates too much temporal error that drowns out the high-order spatial accuracy. Conversely, one that is too small wastes computational resources in a situation where the spatial error dominates. The optimal time step choice can be identified through time-step convergence studies for a particular spatial discretization and time-marching scheme. This approach of "decreasing the

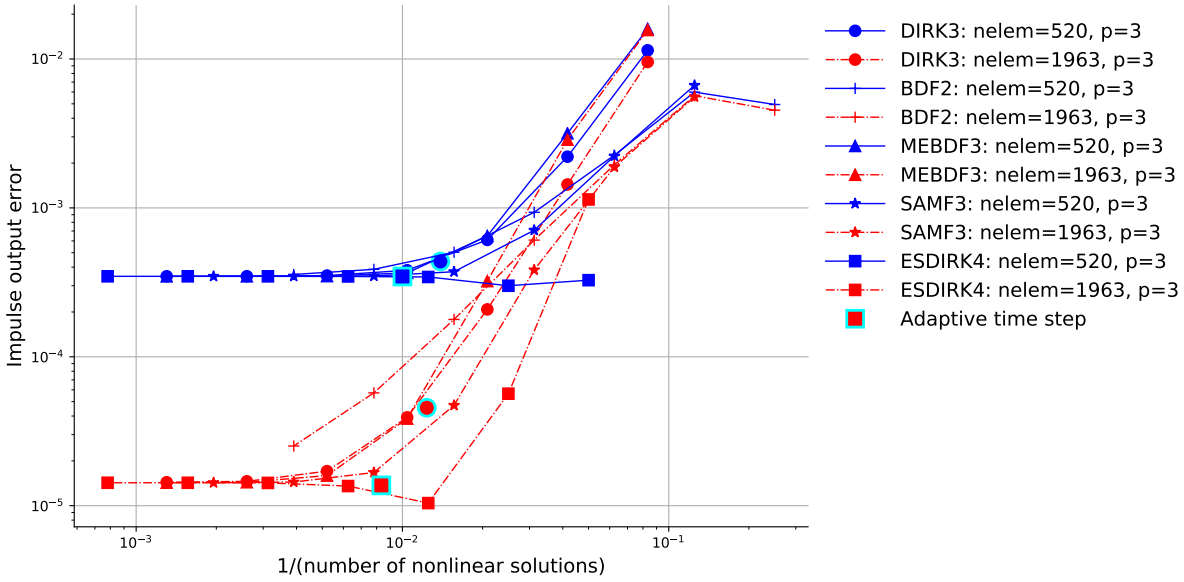


Fig. 16 Airfoil impulse convergence efficiency results, versus total number of nonlinear solutions, for $p = 3$ and t^3 startup.

time step until the answers stops changing" is often employed in practice. We note that for two of the problems studied in this work, the hybrid multistep/multistage split Adams-Moulton method (SAMF3) was found to be the most efficient out of the methods tested, due to its requirement of only two stages for fourth-order accuracy. However, it requires uniform time steps, which makes its application challenging and perhaps less efficient in a variable-time-step setting.

The multiple unsteady simulations required for a convergence study are expensive. Therefore, this work also presents an automatic time-step control strategy for selecting the time step adaptively at each unsteady iteration to balance the spatial and temporal errors. This strategy requires residual evaluations on a spatially finer approximation space, which generally do not add significant cost compared to the nonlinear implicit solutions. No adjoint solutions or backward-in-time integrations are required. The results demonstrate the efficacy of this strategy: the time step is controlled in a manner that makes the spatial and temporal errors balanced, so that only one unsteady simulation is required, with the temporal resolution set and adjusted automatically.

Finally, a more fundamental question raised by the present study is whether the local spatial and temporal errors need to be balanced at all times through the course of the simulation. At times in the solution evolution when the local spatial error becomes very small, the time step may not need to be reduced, since although the temporal error may be larger than the spatial error for that time step, it may still be less than the errors committed on previous, or future, time steps. Pegging the temporal error to the spatial error locally at each time may thus overly restrict the time step. An alternative strategy is to keep track of the average error per time step, and to permit larger time steps as long as the average error, or a fraction of it, is not exceeded. Of course, knowledge of errors in future time steps would still not be available, but this is a reasonable start and improvement over the present local error strategy. This question is a direction of future investigation.

A. ESDIRK coefficients

For the ESDIRK5 method used in this work, the coefficients are

$$a_{ij} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ \frac{8611}{62500} & \frac{-1743}{31250} & \frac{1}{4} & 0 & 0 & 0 \\ \frac{5012029}{34652500} & \frac{-654441}{2922500} & \frac{174375}{388108} & \frac{1}{4} & 0 & 0 \\ \frac{15267082809}{155376265600} & \frac{-71443401}{120774400} & \frac{730878875}{902184768} & \frac{2285395}{8070912} & \frac{1}{4} & 0 \\ \frac{82889}{524892} & 0 & \frac{15625}{83664} & \frac{69875}{102672} & \frac{-2260}{8211} & \frac{1}{4} \end{bmatrix}$$

$$b_i = \begin{bmatrix} 0 & \frac{1}{2} & \frac{83}{250} & \frac{31}{50} & \frac{17}{20} & 1 \end{bmatrix}$$

For the ESDIRK5 method used in this work, the coefficients are[7]

$$a_{ij} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{41}{200} & \frac{41}{200} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{41}{400} & \frac{-567603406766}{11931857280679} & \frac{41}{200} & 0 & 0 & 0 & 0 & 0 \\ \frac{683785636431}{9252920307686} & 0 & \frac{-110385047103}{1367015193373} & \frac{41}{200} & 0 & 0 & 0 & 0 \\ \frac{3016520224154}{10081342136671} & 0 & \frac{30586259806659}{12414158314087} & \frac{-22760509404356}{11113319521817} & \frac{41}{200} & 0 & 0 & 0 \\ \frac{218866479029}{1489978393911} & 0 & \frac{638256894668}{5436446318841} & \frac{-1179710474555}{5321154724896} & \frac{-60928119172}{8023461067671} & \frac{41}{200} & 0 & 0 \\ \frac{1020004230633}{5715676835656} & 0 & \frac{25762820946817}{25263940353407} & \frac{-2161375909145}{9755907335909} & \frac{-211217309593}{5846859502534} & \frac{-4269925059573}{7827059040749} & \frac{41}{200} & 0 \\ \frac{-872700587467}{9133579230613} & 0 & 0 & \frac{22348218063261}{9555858737531} & \frac{-1143369518992}{8141816002931} & \frac{-39379526789629}{19018526304540} & \frac{32727382324388}{42900044865799} & \frac{41}{200} \end{bmatrix}$$

$$b_i = \begin{bmatrix} 0 & \frac{41}{100} & \frac{2935347310677}{11292855782101} & \frac{1426016391358}{7196633302097} & \frac{92}{100} & \frac{24}{100} & \frac{3}{5} & 1 \end{bmatrix}$$

Note that the first, all-zero, row of a_{ij} is never used and is only present for consistency with the row/column indices beginning at zero for ESDIRK methods.

References

[1] Wang, Z., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H., Kroll, N., May, G., Persson, P.-O., van Leer, B., and Visbal, M., "High-Order CFD Methods: Current Status and Perspective," *International Journal for Numerical Methods in Fluids*, Vol. 72, 2013, pp. 811–845. https://doi.org/10.1002/fld.3767.

[2] Fidkowski, K. J., and Darmofal, D. L., "Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics," *AIAA Journal*, Vol. 49, No. 4, 2011, pp. 673–694. https://doi.org/10.2514/1.J050073.

[3] Cockburn, B., and Shu, C.-W., "Runge-Kutta discontinuous Galerkin methods for convection-dominated problems," *Journal of Scientific Computing*, Vol. 16, No. 3, 2001, pp. 173–261. https://doi.org/https://doi.org/10.1023/A:1012873910884.

[4] Richter, T., "Discontinuous Galerkin as Time-Stepping Scheme for the Navier-Stokes Equations," *Fourth International Conference on High Performance Scientific Computing Modeling, Simulation and Optimization of Complex Processes*, Hanoi, Vietnam, 2009.

[5] Fidkowski, K. J., "Output error estimation strategies for discontinuous Galerkin discretizations of unsteady convection-dominated flows," *International Journal for Numerical Methods in Engineering*, Vol. 88, No. 12, 2011, pp. 1297–1322. https://doi.org/10.1002/nme.3224.

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

[7] Kennedy, C. A., and Carpenter, M. H., "Additive Runge-Kutta schemes for convection-diffusion-reaction equations," *Applied Numerical Mathematics*, Vol. 44, 2003, pp. 139–181.

[8] Barth, T. J., "Space-Time Error Representation and Estimation in Navier-Stokes Calculations," *Complex Effects in Large Eddy Simulations*, edited by S. C. Kassinos, C. A. Langer, G. Iaccarino, and P. Moin, Springer Berlin Heidelberg, Lecture Notes in Computational Science and Engineering Vol 26, 2007, pp. 29–48.

[9] Meidner, D., and Vexler, B., "Adaptive Space-Time Finite Element Methods for Parabolic Optimization Problems," *SIAM Journal on Control Optimization*, Vol. 46, No. 1, 2007, pp. 116–142. https://doi.org/https://doi.org/10.1137/060648994.

[10] Mani, K., and Mavriplis, D. J., "Error Estimation and Adaptation for Functional Outputs in Time-Dependent Flow Problems," *Journal of Computational Physics*, Vol. 229, 2010, pp. 415–440. https://doi.org/https://doi.org/10.1016/j.jcp.2009.09.034.

[11] Fidkowski, K. J., and Luo, Y., "Output-based Space-Time Mesh Adaptation for the Compressible Navier-Stokes Equations," *Journal of Computational Physics*, Vol. 230, 2011, pp. 5753–5773. https://doi.org/10.1016/j.jcp.2011.03.059.

[12] Fidkowski, K. J., "An Output-Based Dynamic Order Refinement Strategy for Unsteady Aerodynamics," AIAA Paper 2012-77, 2012. https://doi.org/10.2514/6.2012-77.

[13] Flynt, B. T., and Mavriplis, D. J., "Discrete Adjoint Based Adaptive Error Control in Unsteady Flow Problems," AIAA Paper 2012-0078, 2012.

[14] Kast, S. M., and Fidkowski, K. J., "Output-based Mesh Adaptation for High Order Navier-Stokes Simulations on Deformable Domains," *Journal of Computational Physics*, Vol. 252, No. 1, 2013, pp. 468–494. https://doi.org/10.1016/j.jcp.2013.06.007.

[15] Fidkowski, K. J., "Output-Based Space-Time Mesh Optimization for Unsteady Flows Using Continuous-in-Time Adjoints," *Journal of Computational Physics*, Vol. 341, No. 15, 2017, pp. 258–277. https://doi.org/10.1016/j.jcp.2017.04.005.

[16] Reed, W., and Hill, T., "Triangular Mesh Methods for the Neutron Transport Equation," Los Alamos Scientific Laboratory Technical Report LA-UR-73-479, 1973.

[17] Roe, P., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372. https://doi.org/https://doi.org/10.1016/0021-9991(81)90128-5.

[18] Bassi, F., and Rebay, S., "GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations," *Discontinuous Galerkin Methods: Theory, Computation and Applications*, edited by B. Cockburn, G. Karniadakis, and C.-W. Shu, Springer, Berlin, 2000, pp. 197–208.

[19] Fidkowski, K. J., "High-Order Output-Based Adaptive Methods for Steady and Unsteady Aerodynamics," *37^{th} Advanced CFD Lectures series; Von Karman Institute for Fluid Dynamics (December 9–12 2013)*, edited by H. Deconinck and R. Abgrall, von Karman Institute for Fluid Dynamics, 2013.

[20] Nørsett, S. P., and Thomsen, P. G., "Local error control in SDIRK-methods," *BIT Numerical Mathematics*, Vol. 26, No. 1, 1986, pp. 100–113.

[21] Alexander, R., "Diagonally implicit Runge-Kutta methods for stiff ODE's," *SIAM Journal on Numerical Analysis*, Vol. 14, No. 6, 1977, pp. 1006–1021.

[22] Rattenbury, N., "Almost Runge-Kutta Methods for Stiff and Non-Stiff Problems," Ph.D. thesis, The University of Auckland, 2005.

[23] Cash, J., "The integration of stiff initial value problems in ODEs using modified extended backward differentiation formulae," *Computers & mathematics with applications*, Vol. 9, No. 5, 1983, pp. 645–657. https://doi.org/https://doi.org/10.1016/0898-1221(83)90122-0.

[24] Cash, J., and Considine, S., "An MEBDF Code for Stiff Initial Value Problems," *ACM Transactions on Mathematical Software*, Vol. 18, No. 2, 1992, pp. 142–155. https://doi.org/10.1145/146847.146922.

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

[26] Voss, D. A., and Casper, M. J., "Efficient Split Linear Multistep Methods for Stiff Ordinary Differential Equations," *SIAM Journal on Scientific and Statistical Computing*, Vol. 10, No. 5, 1989, pp. 990–999.

[27] Gustafsson, K., Lundh, M., and Søderlind, G., "A PI stepsize control for the numerical solution of ordinary differential equations," *BIT Numerical Mathematics*, Vol. 28, 1988, pp. 270–287. https://doi.org/10.1007/BF01934091.

[28] Pan, Y., Yan, Z.-G., Peiró, J., and Sherwin, S. J., "Development of a Balanced Adaptive Time-Stepping Strategy Based on an Implicit JFNK-DG Compressible Flow Solver," *Communications on Applied Mathematics and Computation*, Vol. 4, 2022, pp. 728–757. https://doi.org/10.1007/s42967-021-00138-1.

[29] Persson, P.-O., Fidkowski, K. J., and Wukie, N. A., "High-Fidelity CFD Workshop 2022: Mesh Motion," AIAA Paper 2021–1551, 2021. https://doi.org/10.2514/6.2021-1551.

[30] Persson, P.-O., Bonet, J., and Peraire, J., "Discontinuous Galerkin Solution of the Navier-Stokes Equations on Deformable Domains," *Computer Methods in Applied Mechanics and Engineering*, Vol. 198, 2009, pp. 1585–1595.