Journal of Computational Physics 399 (2019) 108913

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Locally adaptive pseudo-time stepping for high-order Flux Reconstruction

N.A. Loppi [a,*], F.D. Witherden [a], A. Jameson [b], P.E. Vincent [a]

[a] Department of Aeronautics, Imperial College London, SW7 2AZ, United Kingdom
[b] Department of Aerospace Engineering, Texas A&M University, 3127 TAMU, TX, USA

ARTICLE INFO

Article history:
Received 8 January 2019
Received in revised form 10 July 2019
Accepted 24 August 2019
Available online 29 August 2019

Keywords:
Dual time stepping
Convergence acceleration
Incompressible flows
Artificial compressibility
Flux reconstruction
Modern hardware

ABSTRACT

This paper proposes a novel locally adaptive pseudo-time stepping convergence acceleration technique for dual time stepping which is a common integration method for solving unsteady low-Mach preconditioned/incompressible Navier-Stokes formulations. In contrast to standard local pseudo-time stepping techniques that are based on computing the local pseudo-time steps directly from estimates of the local Courant-Friedrichs-Lewy limit, the proposed technique controls the local pseudo-time steps using local truncation errors which are computed with embedded pair RK schemes. The approach has three advantages. First, it does not require an expression for the characteristic element size, which are difficult to obtain reliably for curved mixed-element meshes. Second, it allows a finer level of locality for high-order nodal discretisations, such as FR, since the local time-steps can vary between solution points and field variables. Third, it is well-suited to being combined with P-multigrid convergence acceleration. Results are presented for a laminar 2D cylinder test case at $Re = 100$. A speed-up factor of 4.16 is achieved compared to global pseudo-time stepping with an RK4 scheme, while maintaining accuracy. When combined with P-multigrid convergence acceleration a speed-up factor of over 15 is achieved. Detailed analysis of the results reveals that pseudo-time steps adapt to element size/shape, solution state, and solution point location within each element. Finally, results are presented for a turbulent 3D SD7003 airfoil test case at $Re = 60,000$. Speed-ups of similar magnitude are observed, and the flow physics is found to be in good agreement with previous studies.

1. Introduction

Dual time stepping [1] is an integration method in which an unsteady problem in physical time is solved implicitly via a series of steady-state problems in pseudo-time. It was originally proposed to circumvent the strict Courant-Friedrichs-Lewy (CFL) condition associated with explicit solution of the compressible Navier-Stokes equations. Currently, dual time stepping is most commonly used together with unsteady low-Mach preconditioned Navier-Stokes formulations which alter characteristic wave speeds to reduce systems stiffness [2]. One such method is the Artificial Compressibility Method (ACM) of Chorin [3] which was developed in the late 1960s as an alternative to pressure correction based methods for solving steady incompressible fluid flow problems. Rather than projecting the pressure with a Poisson equation, it is solved iteratively by driving a modified continuity equation towards a divergence free solution. Common practice with dual time stepping is that

* Corresponding author.
E-mail address: n.loppi15@imperial.ac.uk (N.A. Loppi).

https://doi.org/10.1016/j.jcp.2019.108913
0021-9991/

the physical time is discretised with a Backward Difference Formula (BDF) scheme, and the pseudo-time problem is solved either explicitly or implicitly.

There have been various attempts to apply the ACM in a high-order context. Bassi [4] first succeeded in applying the approach with a Discontinuous Galerkin (DG) scheme to solve steady incompressible flow problems. Later, Liang et al. [5] extended the method to Spectral Difference (SD) schemes in 2D, providing support for unsteady flows via dual time stepping. Recently, Cox et al. [6] successfully applied the method with Flux Reconstruction (FR) in 2D, and introduced a fully implicit pseudo-time stepping strategy. In the current study, we utilise the cross-platform high-order ACM solver with FR developed by Loppi et al. [7] in the Python-based open-source PyFR framework. The solver supports a range of computer architectures, from laptops to the largest supercomputers, via a platform-unified templating approach that can generate and compile CUDA, OpenCL and C/OpenMP code at runtime. In the solver, pseudo-time stepping is performed explicitly with P-multigrid convergence acceleration. The choice of explicit methods is motivated by current trends in modern hardware; most notably an abundance of compute capability relative to memory bandwidth, extensive parallelism, and low memory per core. Using explicit schemes, the majority of operations can be cast as matrix-matrix multiplications with minimal communication, whereas implicit methods require computation of large flux Jacobians and significant memory indirection due to inter-element couplings.

The efficiency of dual time stepping depends strongly on the rate at which pseudo iterations converge within each physical time step. A popular explicit convergence acceleration technique is local pseudo-time stepping, which allows the pseudo-time step size to vary between elements. The majority of current local pseudo-time stepping approaches are based on identifying the maximum locally stable time step directly from a local CFL number, calculation of which requires an estimate of the local characteristic element size. For unstructured grids the definition of the characteristic element size is often based on heuristics. Two popular definitions are the radius of an inscribed sphere or the distance to the element boundary along a streamline [8,9]. In this study, a new alternative approach based on adaptive error control via embedded pair RK schemes is proposed. The main advantage of this approach is that it does not require computing characteristic element sizes which can be difficult to obtain reliably for high-order curved unstructured grids of mixed element types. In addition, the approach allows a finer level of locality, as the local time-steps can vary between solution points and field variables. Due to its nodal nature, it is also well-suited to work with P-multigrid convergence acceleration, since the pseudo time-step sizes can be locally projected onto different solution bases in the P-multigrid cycle.

The paper is structured as follows. Sections 2 and 3 provide brief summaries on the FR discretisation and relevant time-integration techniques. Section 4 introduces the unsteady artificial compressibility method for computing incompressible flows with FR using dual time stepping. Section 5 details the proposed locally adaptive pseudo-time stepping approach, including P-multigrid acceleration, and introduces an implementation in the open-source PyFR framework. In Section 6, the performance is assessed with a 2D cylinder test case, and the underlying adaptation mechanisms are analysed. In Section 7, the technology is applied to a SD7003 airfoil simulation to demonstrate its utility for turbulent flow problems. Finally, conclusions are drawn in Section 8.

2. Flux Reconstruction

Flux Reconstruction method unifies nodal DG and SD schemes within a single framework. It was first introduced by Hyunh for advection in [10] and for diffusion in [11]. A summary of the FR formulation for mixed unstructured grids is given below. More detailed presentations can be found in [12–14]. Consider a finite solution domain $\mathbf{\Omega}$ in Euclidean space $\mathbb{R}^{N_D}$ with a coordinate system $\mathbf{x} = x_i \in \mathbb{R}^{N_D}$, where N_D is the number of dimensions. A conservation law in the domain takes the form

$$\frac{\partial u_\alpha}{\partial t} = -\nabla \cdot \mathbf{f}_{\boldsymbol{\alpha}} \,, \tag{1}$$

where $u_\alpha = u_\alpha(\mathbf{x}, t)$ is the solution state for a field variable α and $\mathbf{f}_{\boldsymbol{\alpha}} = \mathbf{f}_{\boldsymbol{\alpha}}(u_\alpha, \nabla \mathbf{u}_\alpha)$ is the associated flux. The domain is discretised using a set of suitable element types ε, such as line elements in $N_D = 1$, quadrilaterals and triangles in $N_D = 2$, and hexahedra and tetrahedra in $N_D = 3$. The elements in the discretised domain must conform according to

$$\mathbf{\Omega} = \bigcup_{e \in \varepsilon} \mathbf{\Omega}_e \,, \qquad \mathbf{\Omega}_e = \bigcup_{n=1}^{|\mathbf{\Omega}_e|} \mathbf{\Omega}_{en} \,, \qquad \bigcap_{e \in \varepsilon} \bigcap_{n=1}^{|\mathbf{\Omega}_e|} \mathbf{\Omega}_{en} = \emptyset \,, \tag{2}$$

where the subscript e refers to an element type and $|\mathbf{\Omega}_e|$ is the number of elements of that type.

For efficient numerical implementation, all operations are performed in a transformed space with a coordinate system $\tilde{\mathbf{x}} = \tilde{x}_i \in \mathbb{R}^{N_D}$. Each physical element $\mathbf{\Omega}_e$ is mapped into its respective transformed standard element $\tilde{\mathbf{\Omega}}_e \in \mathbb{R}^{N_D}$ via a vector mapping function $\mathcal{M}_i$ as

$$\tilde{x}_i = \boldsymbol{\mathcal{M}}^{-1}_{eni}(\mathbf{x}) \,, \tag{3}$$

$$x_i = \boldsymbol{\mathcal{M}}_{eni}(\tilde{\mathbf{x}}) \,, \tag{4}$$

where i denotes the spatial component of the mapping function. The Jacobian matrices and determinants associated with the mapping are defined as

$$\mathbf{J}_{en}^{-1} = J_{enij}^{-1} = \frac{\partial \mathcal{M}_{eni}^{-1}}{\partial x_j} , \qquad \mathbf{J}_{en} = J_{enij} = \frac{\partial \mathcal{M}_{eni}}{\partial \tilde{x}_j} , \tag{5}$$

$$\mathcal{J}_{en}^{-1} = \det \mathbf{J}_{en}^{-1} = \frac{1}{\mathcal{J}_{en}} , \qquad \mathcal{J}_{en} = \det \mathbf{J}_{en} , \tag{6}$$

where j denotes the spatial component of the derivative. Using the above relations, the transformed flux and transformed gradient of the solution can be written as a function of the physical solution as

$$\tilde{\mathbf{f}}_{en\alpha}(\tilde{\mathbf{x}}, t) = \mathcal{J}_{en}(\tilde{\mathbf{x}}) \mathbf{J}_{en}^{-1}(\mathcal{M}_{en}(\tilde{\mathbf{x}})) \mathbf{f}_{en\alpha}(\mathcal{M}_{en}(\tilde{\mathbf{x}}), t) , \tag{7}$$

$$\widetilde{\nabla} \boldsymbol{u}_{en\alpha}(\tilde{\mathbf{x}}, t) = \mathbf{J}_{en}^{T}(\tilde{\mathbf{x}}) \nabla \mathbf{u}_{en\alpha}(\mathcal{M}_{en}(\tilde{\mathbf{x}}), t) , \tag{8}$$

where $\widetilde{\nabla} = \partial / \partial \tilde{x}_i$. Moreover, Equation (1) can be expressed in terms of the transformed divergence of the transformed flux as

$$\frac{\partial u_{en\alpha}}{\partial t} = -\mathcal{J}_{en}^{-1} \widetilde{\nabla} \cdot \tilde{\mathbf{f}}_{en\alpha} . \tag{9}$$

In the FR method, piece-wise discontinuous polynomials of order P are used to represent the solution within each element. Take $\{\tilde{\mathbf{x}}_{ei}^{u}\}$ to be a set of solution points associated with a given standard element type, where $0 \leq i < N_e$, with N_e being the number of solution points. Examples of such point distributions are Gauss-Legendre or Gauss-Lobatto points, when $N_D = 1$, Witherden-Vincent points [15] for triangles, when $N_D = 2$, and Shunn-Ham points [16] for tetrahedra, when $N_D = 3$. The set of solution points $\{\tilde{\mathbf{x}}_{ei}^{u}\}$ can be used to define a nodal basis set $\{l_{ei}(\tilde{\mathbf{x}})\}$ of order $P(N_e)$ that spans a polynomial space $\mathcal{P}$ and satisfies the property $l_{ei}(\tilde{\mathbf{x}}_{ej}^{u}) = \delta_{ij}$. Following the methodology in [17], the nodal basis can be formed by first defining any modal basis set $\{L_{ei}(\tilde{\mathbf{x}})\}$ that spans $\mathcal{P}$, and calculating the entries in the associated Vandermonde matrix $\mathcal{V}_{eij} = L_{ei}(\tilde{\mathbf{x}}_{ej}^{u})$. Subsequently, a nodal basis can be constructed as $l_{ei}(\tilde{\mathbf{x}}_{ej}^{u}) = \mathcal{V}_{eij}^{-1} L_{ej}$. In addition to the solution points, a set of element-type-specific flux points $\{\tilde{\mathbf{x}}_{i}^{f}\}$, where $0 \leq i < N_e^f$, with N_e^f being the number of flux points, are defined at the element interfaces. Each element interface, apart from those at the domain boundaries, contain the flux points of two neighbouring elements. For a given flux point pair eij and $e'i'j'$, the mapping function must return the same physical location according to

$$\boldsymbol{\mathcal{M}}_{en}(\tilde{\mathbf{x}}_{ei}^{f}) = \boldsymbol{\mathcal{M}}_{e'n'}(\tilde{\mathbf{x}}_{e'i'}^{f}) . \tag{10}$$

For computing the right-hand-side, the first step is obtaining the discontinuous solution at the flux points from the interpolating solution polynomial as

$$u_{ein\alpha}^{f} = u_{ejn\alpha}^{u} l_{ej}(\tilde{\mathbf{x}}_{ei}^{f}) . \tag{11}$$

These values are used to find a common interface solution at the flux points via a scalar function $\mathcal{C}(u_{\mathrm{L}}, u_{\mathrm{R}})$, where the superscripts R and L denote the right and left states respectively. The common values are assigned as

$$\mathcal{C}u_{ein\alpha}^{f} = \mathcal{C}u_{e'i'n'\alpha}^{f} = \mathcal{C}(u_{ein\alpha}^{f}, u_{e'i'n'\alpha}^{f}) . \tag{12}$$

The second step is to compute the gradient of the solution at the solution points. For this purpose, we form a vector correction function $\mathbf{g}_{ei}^{f}(\tilde{\mathbf{x}})$ which satisfies

$$\hat{\tilde{\mathbf{n}}}_{ej} \cdot \mathbf{g}_{ei}^{f}(\tilde{\mathbf{x}}_{ej}^{f}) = \delta_{ij} , \tag{13}$$

where $\hat{\tilde{\mathbf{n}}}_{ej}$ is the outward facing normal vector. This allows the transformed gradient of the solution at the solution points to be computed as

$$\widetilde{\nabla} \boldsymbol{u}_{ein\alpha}^{u} = \left(\hat{\tilde{\mathbf{n}}} \widetilde{\nabla}\right)_{ej} \cdot \mathbf{g}_{ej}^{f}(\tilde{\mathbf{x}}_{ei}^{u}) \{\mathcal{C}u_{ejn\alpha}^{f} - u_{ejn\alpha}^{f}\} + u_{ekn\alpha}^{u} \widetilde{\nabla} l_{ek}(\tilde{\mathbf{x}}_{ei}^{u}) , \tag{14}$$

which transforms into physical space as

$$\nabla \mathbf{u}_{ein\alpha}^{u} = \mathbf{J}_{ein}^{-T} \widetilde{\nabla} \boldsymbol{u}_{ein\alpha}^{u} , \tag{15}$$

where $\mathbf{J}_{ein}^{-T} = \mathbf{J}_{en}^{-T}(\tilde{\mathbf{x}}_{ei})$. As a third step, the transformed flux at the solution points is evaluated as

$$\tilde{\mathbf{f}}_{ein\alpha}^{u} = \mathcal{J}_{ein} \mathbf{J}_{ein}^{-1} \mathbf{f}_{\alpha}(u_{ein\alpha}^{u}, \nabla \mathbf{u}_{ein\alpha}^{u}) , \tag{16}$$

and the normal component of the transformed flux at the flux points as

$$\tilde{f}^{f\perp}_{ein\alpha} = l_{ej}(\tilde{\mathbf{x}}^{f}_{ei})\hat{\tilde{\mathbf{n}}}_{ei} \cdot \tilde{\mathbf{f}}^{u}_{ejn\alpha} \, . \tag{17}$$

The fourth step is to compute the common normal fluxes at the flux point pairs with a function $\mathcal{F}_\alpha(u_\mathrm{L}, \nabla\mathbf{u}_\mathrm{L}, u_\mathrm{R}, \nabla\mathbf{u}_\mathrm{R}, \hat{\mathbf{n}})$, that comprises of performing a Riemann solve for the inviscid part and using the LDG approach for the viscous part. The common values are assigned as

$$\mathcal{F}_\alpha f^{f\perp}_{ein\alpha} = -\mathcal{F}_\alpha f^{f\perp}_{e'i'n'\alpha} = \mathcal{F}_\alpha(u^{f}_{ein}, \nabla\mathbf{u}^{f}_{ein}, u^{f}_{e'i'n'}, \nabla\mathbf{u}^{f}_{e'i'n'}, \hat{\mathbf{n}}_{ein}) \, , \tag{18}$$

where

$$\nabla\mathbf{u}^{f}_{ein\alpha} = l_{ejn}(\tilde{\mathbf{x}}^{f}_{ei})\nabla\mathbf{u}^{u}_{ejn\alpha} \, , \tag{19}$$

and they are transformed into the standard element space via

$$\mathcal{F}_\alpha \tilde{f}^{f\perp}_{ein\alpha} = \mathcal{J}^{f}_{ein} n_{ein} \mathcal{F}_\alpha f^{f\perp}_{ein\alpha} \, , \tag{20}$$

with n_{ein} being the magnitude of the physical normal at a flux point. Finally, the divergence of the continuous flux can be computed with a procedure analogous to Equation (14), as

$$\left(\widetilde{\nabla}\cdot\tilde{\mathbf{f}}\right)^{u}_{ein\alpha} = \left[\widetilde{\nabla}\cdot\mathbf{g}^{f}_{ej}(\tilde{\mathbf{x}}^{u}_{ei})\{\mathcal{F}_\alpha\tilde{f}^{f\perp}_{ejn\alpha} - \tilde{f}^{f\perp}_{ejn\alpha}\} + \tilde{\mathbf{f}}^{u}_{ekn\alpha}\cdot\widetilde{\nabla} l_{ek}(\tilde{\mathbf{x}}^{u}_{ei})\right] . \tag{21}$$

This serves as the discrete representation of the flux divergence, and the solution can be in advanced in time by integrating

$$\frac{\partial u_{ein\alpha}}{\partial t} = -\left(\mathcal{J}^{-1}\right)^{u}_{ein}\left(\widetilde{\nabla}\cdot\tilde{\mathbf{f}}\right)^{u}_{ein\alpha} \, . \tag{22}$$

3. Time-integration

3.1. Explicit Runge-Kutta schemes

Consider an ODE of the form

$$\frac{du}{dt} = g(t,u) \, . \tag{23}$$

Explicit Runge-Kutta (RK) methods for solving Equation (23) can be written in the general form

$$u^{n+1} = u^n + \Delta t \sum_{i=1}^{s} b_i k_i \, , \tag{24}$$

$$k_i = g(t + c_i\Delta t, u^n + \Delta t \sum_{j=1}^{i-1} a_{ij}k_j) \, , \tag{25}$$

where s is the stage count, $\mathbf{A} = a_{ij}$ is an $s\times s$ strictly lower triangular matrix, and $\mathbf{b} = b_i$ and $\mathbf{c} = c_i$ are vectors of length s. The coefficients of a generalised explicit RK scheme are typically presented in a Butcher tableau [18]

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b} \end{array} . \tag{26}$$

The constraints for an explicit RK scheme are

$$c_i = \sum_{j=1}^{s} a_{ij} \, , \tag{27}$$

$$\sum_{i=1}^{s} b_i = 1 \, , \tag{28}$$

$$a_{ij} = 0, \quad j \geq i \, . \tag{29}$$

By applying an RK scheme to a linear test problem $g(t,u) = \omega u$, where $\omega \in \mathbb{C}$, the stability polynomial of an explicit RK scheme can be defined directly from the Butcher tableau as

$$P^{(s,q)}(z) = 1 + z\mathbf{b}^T(\mathbf{I} - z\mathbf{A})^{-1}\mathbf{e} = \frac{|\mathbf{I} - z\mathbf{A} + z\mathbf{e}\mathbf{b}^T|}{|\mathbf{I} - z\mathbf{A}|} \, . \tag{30}$$

Table 1
Butcher tableau of the RK3(2)4[2R+] embedded pair represented as (a) rational coefficients and (b) real coefficients, where the first rows of **b** are the $b_i^{(4,3)}$ coefficients resulting in third order temporal accuracy and the second rows are the $b_i^{(4,2)}$ coefficients resulting in second order accuracy.

(a)

0				
c_2	$\frac{11847461282814}{36547543011857}$			
c_3	$\frac{1017324711453}{9774461848756}$	$\frac{3943225443063}{7078155732230}$		
c_4	$\frac{1017324711453}{9774461848756}$	$\frac{8237718856693}{13685301971492}$	$-\frac{346793006927}{4029903576067}$	
$b_i^{(4,3)}$	$\frac{1017324711453}{9774461848756}$	$\frac{8237718856693}{13685301971492}$	$\frac{57731312506979}{19404895981398}$	$-\frac{101169746363290}{37734290219643}$
$b_i^{(4,2)}$	$\frac{15763415370699}{46270243929542}$	$\frac{514528521746}{5659431552419}$	$\frac{27030193851939}{9429696342944}$	$-\frac{69544964788955}{30262026368149}$

$c_i = A_{i,i-1} + \sum_{j=1}^{i-2} + b_{j-2}^{(4,3)},\ 2 \le i \le 4$

(b)

0				
c_2	0.3241657388287			
c_3	0.1040798692751	0.557097864506		
c_4	0.1040798692751	0.601939136882	−0.086054914313	
$b_i^{(4,3)}$	0.104079869275	0.601939136882	2.975090026884	−2.681109033041
$b_i^{(4,2)}$	0.340681484081	0.090915230086	2.866496742725	−2.298093456893

$c_i = A_{i,i-1} + \sum_{j=1}^{i-2} + b_{j-2}^{(4,3)},\ 2 \le i \le 4$

where $z = \omega \Delta t$, **e** is a vector of ones, and the superscript q denotes the temporal order of the scheme. The stability region S of the RK scheme is

$$S = \left\{ z \in \mathbb{C} : |P_{s,p}| \le 1 \right\} , \tag{31}$$

with the linear stability condition being

$$\Delta t \omega \subseteq S . \tag{32}$$

Embedded RK pairs are a family of RK schemes of consecutive temporal orders q and $q-1$ for which **A** are equal. This property allows the temporal truncation error of the embedded pair to be estimated as

$$\xi(t + \Delta t) = \Delta t \sum_{i=1}^{s} (b_i^{(s,q)} - b_i^{(s,q-1)}) k_i . \tag{33}$$

For a given truncation error tolerance, the time-step size can be controlled such that the scheme remains stable. This procedure will be detailed in Section 5. The Butcher tableau of the RK3(2)4[2R+] scheme [19] used in the numerical experiments is shown in Table 1. The naming convention follows that of [19]. The first digit denotes the temporal order-of-accuracy of the higher order scheme followed by the temporal order-of-accuracy of the lower order scheme in parentheses. The third digit is the number of stages and the final string in square brackets describes the number of registers required when the scheme is implemented using the van der Houwen formulation [19].

3.2. Implicit Backward Difference Formula schemes

BDF schemes are a range of implicit multistep methods that can be formulated as

$$u^{n+1} = -\sum_{i=0}^{s-1} B_{i+1} u^{n-i} + \Delta t B_0 g(t^{n+1}, u^{n+1}) , \tag{34}$$

where B_i are the BDF-coefficients. Table 2 shows the BDF-coefficients for the most commonly used schemes. The stability polynomial of BDFs can be defined as

$$P(z) = -\sum_{i=0}^{s-1} B_{i+1} \xi^i + (1 - B_0 z) \xi^s . \tag{35}$$

The stability condition is that all roots of the stability polynomial lie in the region $\left\{ z \in \mathbb{C} : |\xi| \le 1 \right\}$. The BDF schemes are A-stable up to 2nd order. With higher order BDFs, two unstable regions form near the real axis. However, these regions are very small for the third order accurate BDF3, meaning it is still stable for a wide range of z. Moreover, it has been successfully applied to unsteady flow simulations [20].

Table 2
Coefficients for the BDFs.

	B_0	B_1	B_2	B_3
Backward-Euler	1	-1	–	–
BDF2	$\frac{2}{3}$	$-\frac{4}{3}$	$\frac{1}{3}$	–
BDF3	$\frac{6}{11}$	$-\frac{18}{11}$	$\frac{9}{11}$	$-\frac{2}{11}$

4. Unsteady Artificial Compressibility Method with dual time stepping

In the ACM formulation of the incompressible Navier-Stokes equations, the conservative variables in three dimensions are

$$u_\alpha = \begin{Bmatrix} p \\ v_x \\ v_y \\ v_z \end{Bmatrix} , \tag{36}$$

where p is the pressure and v_x, v_y, and v_z are the velocity components in the x, y, and z directions, respectively. The fluxes are defined as

$$\mathbf{f}_{\boldsymbol{\alpha}} = \begin{Bmatrix} f^{\mathrm{e}}_{\alpha x} - f^{\mathrm{v}}_{\alpha x} \\ f^{\mathrm{e}}_{\alpha y} - f^{\mathrm{v}}_{\alpha y} \\ f^{\mathrm{e}}_{\alpha z} - f^{\mathrm{v}}_{\alpha z} \end{Bmatrix} , \tag{37}$$

where

$$f^{\mathrm{e}}_{\alpha x} = \begin{Bmatrix} \zeta v_x \\ v_x^2 + p \\ v_x v_y \\ v_x v_z \end{Bmatrix} , f^{\mathrm{e}}_{\alpha y} = \begin{Bmatrix} \zeta v_y \\ v_y v_x \\ v_y^2 + p \\ v_y v_z \end{Bmatrix} , f^{\mathrm{e}}_{\alpha z} = \begin{Bmatrix} \zeta v_z \\ v_z v_x \\ v_z v_y \\ v_z^2 + p \end{Bmatrix} , \tag{38}$$

$$f^{\mathrm{v}}_{\alpha x} = \nu \begin{Bmatrix} 0 \\ \frac{\partial v_x}{\partial x} \\ \frac{\partial v_y}{\partial x} \\ \frac{\partial v_z}{\partial x} \end{Bmatrix} , f^{\mathrm{v}}_{\alpha y} = \nu \begin{Bmatrix} 0 \\ \frac{\partial v_x}{\partial y} \\ \frac{\partial v_y}{\partial y} \\ \frac{\partial v_z}{\partial y} \end{Bmatrix} , f^{\mathrm{v}}_{\alpha z} = \nu \begin{Bmatrix} 0 \\ \frac{\partial v_x}{\partial z} \\ \frac{\partial v_y}{\partial z} \\ \frac{\partial v_z}{\partial z} \end{Bmatrix} , \tag{39}$$

with ζ being the artificial compressibility relaxation factor and ν the kinematic viscosity.

The ACM formulation of the incompressible Navier-Stokes equations is hyperbolic in nature, allowing artificial pressure waves with finite speeds. These waves distribute the pressure and disappear in the limit of pseudo steady state. Eigenvalues of the inviscid flux Jacobian matrices

$$\mathbf{J}_i = \frac{\partial f^{\mathrm{e}}_{\alpha i}}{\partial u} , \tag{40}$$

are

$$\boldsymbol{\lambda}_i = \{ v_i - c_i,\ v_i,\ v_i,\ v_i + c_i \} , \tag{41}$$

where $c_i = \sqrt{v_i^2 + \zeta}$ is the pseudo speed of sound, and $i \in \{x\ y\ z\}$. The pseudo speed of sound monotonically decreases with decreasing artificial compressibility relaxation factor ζ which can be adjusted to reduce the system stiffness. The common interface flux in this study is computed as

$$\mathcal{F}_\alpha(u_{\mathrm{L}}, \boldsymbol{\nabla}\mathbf{u}_{\mathrm{L}}, u_{\mathrm{R}}, \boldsymbol{\nabla}\mathbf{u}_{\mathrm{R}}, \hat{\mathbf{n}}) = \mathcal{F}^{\mathrm{e}}_\alpha(u_{\mathrm{L}}, u_{\mathrm{R}}, \hat{\mathbf{n}}) - \mathcal{F}^{\mathrm{v}}_\alpha(u_{\mathrm{L}}, \boldsymbol{\nabla}\mathbf{u}_{\mathrm{L}}, u_{\mathrm{R}}, \boldsymbol{\nabla}\mathbf{u}_{\mathrm{R}}, \hat{\mathbf{n}}) , \tag{42}$$

where $\mathcal{F}^{\mathrm{e}}_\alpha$ is the Rusanov Riemann solver [21] and $\mathcal{F}^{\mathrm{v}}_\alpha$ together with $\mathcal{C}(u_{\mathrm{L}}, u_{\mathrm{R}})$ are defined according to the local discontinuous Galerkin (LDG) approach in [17].

The dual time stepping formulation applied to the artificial compressibility equations can be written as

$$\frac{\partial u_\alpha}{\partial \tau} = -\left(\mathbf{I}_c \frac{\partial u_\alpha}{\partial t} + \nabla \cdot \mathbf{f}_{\boldsymbol{\alpha}} \right) , \tag{43}$$

where $\partial u_\alpha / \partial \tau$ is a pseudo time derivative which is marched towards zero with a Runge-Kutta scheme, and $\mathbf{I}_c \partial u_\alpha / \partial t$ is the physical time derivative which is discretised using a BDF scheme. A cancellation matrix $\mathbf{I}_c = \mathrm{diag}\{0\ 1\ 1\ 1\}$ is employed as

a coefficient for the physical time derivative to eliminate it from the continuity equation, forcing the solution to be driven towards a divergence free state according to

$$\lim_{\tau\to\infty}\left[\frac{\partial v_x}{\partial x}+\frac{\partial v_y}{\partial y}+\frac{\partial v_z}{\partial z}\right]=0\,. \tag{44}$$

When the solution is integrated with respect to pseudo time, the physical time derivative can be considered as a source term for the right-hand side that is updated at every step. For example, with a BDF2 discretisation the equations for performing a single pseudo-time step becomes

$$u_\alpha^{n+1,m+1}=u_\alpha^{n+1,m}+\sum_{i=1}^{s}\frac{\Delta\tau b_i}{\alpha_{\mathrm{PI}}}k_i\,, \tag{45}$$

$$k_i=-\nabla\cdot\mathbf{f}_\alpha(u_\alpha^{n+1,m}+\Delta\tau\sum_{j=1}^{i-1}a_{ij}k_i)-\frac{\mathbf{I}_c}{\Delta t B_0}\left(u_\alpha^{n+1,m}+B_1u_\alpha^{n}+B_2u_\alpha^{n-1}\right)\,, \tag{46}$$

where $\Delta\tau$ is the pseudo-time step, n is the physical time-level counter, m is the pseudo-time level counter, and $\alpha_{\mathrm{PI}}=1+b_i\Delta\tau/B_0\Delta t$ is a scaling coefficient which limits $\Delta\tau$ if $\Delta\tau\gg\Delta t$. With explicit pseudo-time steppers this is rarely the case and $\alpha_{\mathrm{PI}}=1$ is enforced in this study.

After each pseudo-step, m is incremented by one. Pseudo-steps are repeated until the solution in physical time is deemed to have converged. Convergence criteria include that the point-wise L^2, or L^∞ norm of the residuals are satisfied up to a given tolerance, or a maximum number of pseudo-time steps have been exceeded. After each pseudo-time cycle, m is restarted from zero, and n is incremented by one.

5. Locally adaptive pseudo-time stepping

5.1. Overview

The efficiency of the dual time stepping algorithm depends on fast convergence of the pseudo steady-state process. The disadvantage of using explicit schemes in pseudo-time is that their stability region is limited by the CFL number making them inefficient at eliminating low frequency error modes on fine grids [22]. The issue can be addressed by adopting implicit schemes to allow much larger pseudo-time steps [6]. However, the trade-off with implicit schemes is that their memory bandwidth and storage requirements are significantly higher due to the flux Jacobian matrices. This is known to be a challenge especially for modern hardware architectures, such as GPUs, that are characterised by an abundance of compute capability relative to memory bandwidth, extensive parallelism, and low memory per core. In addition, many methods for solving the resulting linear system, such as LU-SGS or ILU-GMRES, are not scale invariant and increasing the amount of parallelism tends to reduce their numerical effectiveness.

Local pseudo-time stepping is a widely-used technique for accelerating steady-state convergence explicitly [23,24,2]. The majority of current local pseudo-time stepping approaches compute the local pseudo-time step for element n of type e from the hyperbolic CFL criterion as

$$\Delta\tau_{en}<\frac{h_{en}}{\max(|\lambda_i|_{en})}\,, \tag{47}$$

where h_{en} is the characteristic length of the element, and $\max(|\lambda_i|_{en})$ is the local spectral radius of the inviscid flux Jacobian. For viscous dominated flows it is also necessary to consider the parabolic CFL limit defined as

$$\Delta\tau_{en}<\frac{h_{en}^2}{2\mu}\,, \tag{48}$$

where $\mu=\rho\nu$, with ρ being the density for compressible Navier-Stokes formulations. Definition of h is often based on heuristics and it can have a considerable effect on the performance and accuracy of local time stepping as shown in [25]. With unstructured grids two popular definitions are the radius of an inscribed sphere or the distance to the element boundary along a streamline [8,9].

5.2. Methodology

As an alternative to the current approaches, a local pseudo-time stepping technique based on adaptive error control with embedded pair RK schemes is proposed. The main advantage of the technique is that it does not require an expression for the characteristic element size which is difficult to obtain reliably for curved mixed-element meshes. It also allows a finer level of locality for high-order nodal discretisations, such as FR, since the local time-steps can vary between solution points

and field variables. Additionally, it is well-suited work with P-multigrid convergence acceleration since the pseudo-time steps can be locally projected onto different solution bases.

Embedded pair RK schemes give two approximations of the solution at the next time level, each depending on a different set of b coefficients. In pseudo-time, the difference between these two approximations, often called a truncation error, can be approximated as

$$\xi^u_{ejn\alpha}(\tau + \Delta\tau^u_{ejn\alpha}) = \Delta\tau^u_{ejn\alpha} \sum_{i=1}^{s} (b_i^{(s,q)} - b_i^{(s,q-1)}) k^u_{iejn\alpha} \tag{49}$$

for a field variable α at solution point j within element n of type e. In the current study, the local pseudo-time steps $\Delta\tau^u_{ejn\alpha}$ will be controlled such that the truncation error remains small, and the method remain stable. Specifically, a normalised error

$$\sigma^u_{ejn\alpha} = \frac{\xi^u_{ejn\alpha}(\tau + \Delta\tau^u_{ejn\alpha})}{\kappa} , \tag{50}$$

is kept close to unity, where κ is an error tolerance. With a PI-controller the pseudo-time step adjustment factor is computed as

$$(f_{\text{PI}})^u_{ejn\alpha} = f_{\text{safe}} \left[\sigma^u_{ejn\alpha}\right]^{\phi/q} \left[(\sigma_{\text{prev}})^u_{ejn\alpha}\right]^{-\chi/q} , \tag{51}$$

where ϕ, are χ are PI-controller parameters, f_{safe} is a safety factor, and the subscript prev denotes the error computed at the previous pseudo-time step. For improved robustness, the pseudo-time step adjustment factor is limited by minimum and maximum adjustment factors $f_{\min}$ and $f_{\max}$ as

$$f^u_{ejn\alpha} = \min(f_{\max}, \max(f_{\min}, (f_{\text{PI}})^u_{ejn\alpha})) . \tag{52}$$

The new pseudo-time step is computed as

$$(\Delta\tau_{\text{new}})^u_{ejn\alpha} = \min(\Delta\tau_{\max}, \max(\Delta\tau_{\min}, f^u_{ejn\alpha}\Delta\tau^u_{ejn\alpha})) , \tag{53}$$

with absolute minimum and maximum pseudo-time step limits $\Delta\tau_{\min}$ and $\Delta\tau_{\max}$, and where $\Delta\tau_{\min}$ also serves as an initial guess for the pseudo-time step sizes. In this work, the maximum pseudo-time step limit is defined as $\Delta\tau_{\max} = f_\tau \Delta\tau_{\min}$, where f_τ is the maximum factor by which the pseudo-time steps are allowed to grow.

The approach differs from physical time-step adaptation in several ways. First, in physical time-step adaptation the PI-controller controls only a single global physical time-step value which is computed from the point-wise error with a norm, such as L^2 or L^∞, over all element types, elements, solution points, and field variables. In locally adaptive pseudo-time stepping, all operations are kept completely local and independent for each element type, element, solution point, and field variable. Second, in physical time-step adaptation it is common practice to reject steps that violate error tolerance requirements, and retake them with a smaller step size. In locally adaptive pseudo-time stepping, retaking steps is unnecessary as long the they remain stable because time accuracy is not required.

In all, the locally adaptive pseudo-time stepping approach is parameterised by eight quantities, κ, ϕ, χ, f_{safe}, $f_{\min}$, $f_{\max}$, $\Delta\tau_{\min}$ and f_τ. Appropriate selection of κ, ϕ, χ, f_{safe}, $f_{\min}$ and $f_{\max}$ is based on the characteristics of the controller rather than specifics of a particular simulation. Values of $\phi = 0.4$, $\chi = 0.7$, $\kappa = 10^{-6}$, and $f_{\text{safe}} = 0.8$ are used throughout the study and they are the default values in PyFR. They have been found empirically to work well, and are similar to those used in global physical time-step control [26,27,18]. The minimum and maximum adjustment factors $f_{\min}$ and $f_{\max}$ should be set conservatively e.g. $f_{\min} = 0.98$ and $f_{\max} = 1.01$ to ensure smooth variation of the local pseudo-time step sizes.

5.3. Combining with P-multigrid

To further accelerate the pseudo-steady-state convergence, error-estimate-based locally adaptive pseudo-time stepping can be coupled with a P-multigrid [22,5]. The P-multigrid technique (perhaps better referred to as multi-P) attempts to circumvent the strict CFL limit when solving the pseudo-time problem explicitly by correcting the solution at different polynomial levels. Without altering the computational grid, low order representations of the solution, which exhibit a less restrictive CFL limit, can be used to propagate information with a larger $\Delta\tau$. Another property of P-multigrid is that when the solution is projected to a lower order space, the low frequency error modes appear as higher frequencies respective to the resolution, which allows explicit steppers to eliminate them more effectively.

Initially, two different approaches for combining P-multigrid were studied. The first approach let all P-multigrid levels perform the pseudo-time step control independently of each other. Despite its potential for being fully automated, it lacked robustness in finding the local pseudo-time steps at the intermediate levels and was therefore discarded. The second approach that was studied performs the pseudo-time step control only at the highest level and the local pseudo-time steps are

projected onto different bases locally. Additionally, the projected values at lower levels are increased with a user-specified scaling factor due to less restrictive CFLs. The second approach was found to be superior in terms of robustness, and it also has a smaller memory footprint and computational overhead, since only the highest level performs the pseudo-time step adaptation.

Dual time stepping with P-multigrid follows the procedures detailed in Section 4 with the difference that an additional source term arising from a lower order representation of the solution is added to Equation (43). The spatially discrete form of the governing equation is

$$\frac{\partial u^u_{einl\alpha}}{\partial \tau} = R^u_{einl\alpha} - r^u_{einl\alpha} \,, \tag{54}$$

where

$$R^u_{einl\alpha} = -\left(\mathcal{J}^{-1}\right)^u_{einl} \left(\widetilde{\nabla}\cdot\tilde{\mathbf{f}}\right)^u_{einl\alpha} - \mathbf{I}_c \frac{\partial u^u_{einl\alpha}}{\partial t} \,, \tag{55}$$

with l denoting the P-multigrid level and $r_{einl\alpha}$ being the additional source term which is zero when $l = P$. A single P-multigrid V-cycle with locally adaptive pseudo-time stepping at the highest level can be performed according to Algorithm 1. The P-multigrid cycle should be set such that it goes down to $P = 0$, and the number of smoothing iterations should be increased at $l = 0$ and at $l = P$ during post-cycle smoothing.

In this work, the restriction operator I_r is constructed using the generalised Vandermonde matrix as

$$I_r = \alpha_\tau \mathcal{V}^T_{el'} \tilde{I} \mathcal{V}^{-T}_{el} \,, \tag{56}$$

where $\mathcal{V}_{el'} = L_{eil'}(\tilde{\mathbf{x}}^u_{ejl'})$, $\mathcal{V}_l = L_{eil}(\tilde{\mathbf{x}}^u_{lej})$, with l' denoting the level onto which the solution is projected, and $\tilde{I}$ is an non-square matrix of zeros, except on its main diagonal, where it has entries of one. The leading coefficient α_τ is a pseudo-time step expansion factor, which is unity except when applying the restriction operator to local pseudo-time steps. In those cases it takes a value greater than unity to increase the local pseudo-time steps sizes at lower P-multigrid levels. The approach corresponds to transforming the nodal solution into a modal representation, removing the highest order mode, and transforming back to a nodal basis. Prolongation is simply performed as Lagrangian interpolation according to

$$I_p = l_{eil}(\tilde{\mathbf{x}}^u_{ejl'}) \,. \tag{57}$$

Algorithm 1 A single P-multigrid V-cycle with locally adaptive pseudo-time stepping polynomial between levels P and $l_{\min}$, where $Niters_l$ denotes the number of smoothing iterations at level l. All indices apart from the level index have been dropped for simplicity.

for $l \in \{P, ..., l_{min}+1\}$ **do**
  if l=P **then**
    for $i \in \{0, ..., Niters_l\}$ **do**
      Pseudo-step Equation (54) to update u_l
      Update $\Delta\tau_l$ with locally adaptive pseudo-time stepping
    end
  else
    for $i \in \{0, ..., Niters_l\}$ **do**
      Pseudo-step Equation (54) to update u_l
    end
  end
  Calculate residual defect $d_l = r_l - R(u_l)$
  Restrict solution $u^0_{l-1} = I_r(u_l)$ and store it
  Restrict defect $d_{l-1} = I_r(d_l)$
  Restrict pseudo-time steps $\Delta\tau_{l-1} = I_r(\Delta\tau_l)$ with $\alpha_\tau > 1$
  Evaluate source $r_{l-1} = R(u^0_{l-1}) + d_{l-1}$
end
for $l \in \{l_{min}, ..., P\}$ **do**
  for $i \in \{0, ..., Niters_l\}$ **do**
    Pseudo-step Equation (54) to update u_l
  end
  Calculate correction $\Delta_l = u^0_l - u_l$
  Prolongate correction $\Delta_{l+1} = I_p(\Delta_l)$
  Add correction $u_{l+1} = u_{l+1} + \Delta_{l+1}$
end
for $i \in \{0, ..., Niters_P\}$ **do**
  Pseudo-step Equation (54) to update u_P
  Update $\Delta\tau_P$ with locally adaptive pseudo-time stepping
end

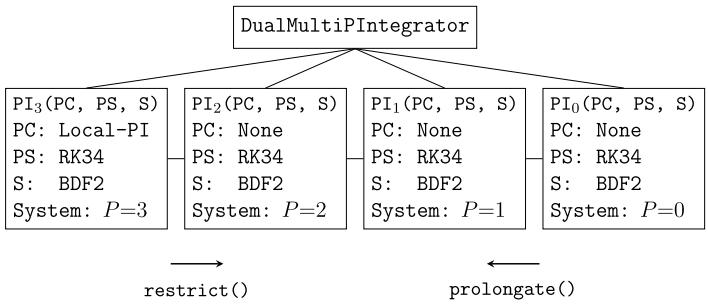

Fig. 1. The structure of P-multigrid with locally adaptive pseudo-time stepping, consisting of an `DualMultiPIntegrator` class that launches several `PseudoIntegrator`s that comprise of a `PseudoController`, `Stepper`, and `PseudoStepper`. RK34 abbreviates the RK3(2)4[2R+] scheme.

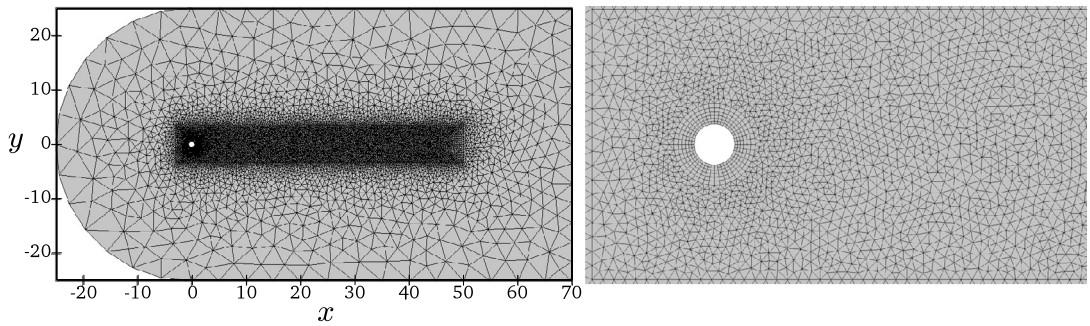

Fig. 2. The computational grid used in the 2D cylinder simulations (left). A zoom view close to the cylinder (right).

5.4. Implementation

PyFR v1.8.0 implements locally adaptive pseudo-time stepping with P-multigrid acceleration in a highly customisable manner. The structure of the implementation is illustrated in Fig. 1 for a 4-level P-multigrid with locally-adaptive pseudo-time-stepping control only at the highest level. The implementation is structured around a `DualMultiPIntegrator` Python class which initialises several `PseudoIntegrator` (`PI`) classes, one for each P-multigrid level. Each `PseudoIntegrator` is a composite of a `PseudoController` (`PC`), which sets the pseudo-time step control approach, a `PseudoStepper` (`PS`), which sets the pseudo-time stepping scheme, and a `Stepper` (`S`), which sets the physical-time stepping scheme. In addition each `PseudoIntegrator` initialises an instance of a `System` class which can compute $\widetilde{\nabla} \cdot \widetilde{\mathbf{f}}$ at each level. The implementation approach allows all `PI`s to advance their solution independently using an arbitrary `PC` and `PS` configuration. Communication between `PI`s only occurs during the restriction and prolongation operations which also automatically project the pseudo-time step field to the correct space.

6. 2D cylinder at $Re = 100$

6.1. Problem specification

The performance of locally adaptive pseudo-time stepping was studied with a 2D cylinder test case at $Re = 100$, based on the cylinder diameter and free-stream velocity. A total of four simulations were performed: a reference Case 1 without any convergence acceleration using RK4 pseudo-time stepping, Case 2 with locally adaptive RK3(2)4[2R+] pseudo-time stepping, Case 3 with P-multigrid acceleration and RK4 smoothing, and Case 4 with P-multigrid acceleration and locally adaptive RK3(2)4[2R+] smoothing. All simulations in the study were performed with a patched version of PyFR v1.8.0. The patch, configuration files and the mesh are provided as Electronic Supplementary Material.

Fig. 2 shows the mixed unstructured computational grid used in the simulations. With the cylinder diameter being 1, the domain consists of a half-circular inflow section with a diameter of 50, and a rectangular wake section with a length of 70. The boundary layer mesh is an O-grid with a height of 0.5 containing 264 quadrilateral elements, whereas the rest of the domain was meshed with 19,174 triangles. All boundary elements are quadratically curved. A Riemann-invariant-based boundary condition, detailed in Appendix A, was prescribed at all far-field boundaries with $u_\infty = \{1\ 1\ 0\}$ and $\zeta_\infty = 100$. The boundary condition was found to be substantially less reflective than conventional BCs used in [5,6]. A no-slip condition was prescribed on the cylinder surface. The initial condition at $t = 0$ was set as $u = \{1\ 1\ 0\}$. All cases were run with $P = 4$ and $\zeta = 4$ on a single Nvidia P100 GPU using double precision. They all discretised the physical time with a BDF2 scheme, where $\Delta t = 2.5 \times 10^{-2}$. To ensure impartial performance comparisons, the fixed pseudo-time step size $\Delta\tau = 4 \times 10^{-4}$ for the reference Case 1 was found by optimising the convergence via manual bisection. In Cases 3 and 4, P-multigrid convergence

Table 3
Summary of the 2D cylinder cases at $Re = 100$, where PS denotes the pseudo-stepper, LAPTS denotes locally adaptive pseudo-time stepping, P-MG denotes P-multigrid, WT denotes the wall-time, and SUF denotes the speed-up factor relative to Case 1.

	PS	LAPTS	P-MG	f_τ	α_τ	WT	SUF	$\overline{C}_D$	St
Case 1	RK4	Off	Off	–	–	13:22:23	1.00	1.339	0.166
Case 2	RK3(2)4[2R+]	On	Off	8.0	–	03:12:53	4.16	1.339	0.166
Case 3	RK4	Off	On	–	2.0	02:07:57	6.27	1.339	0.166
Case 4	RK3(2)4[2R+]	On	On	8.0	1.6	00:52:40	15.24	1.339	0.166

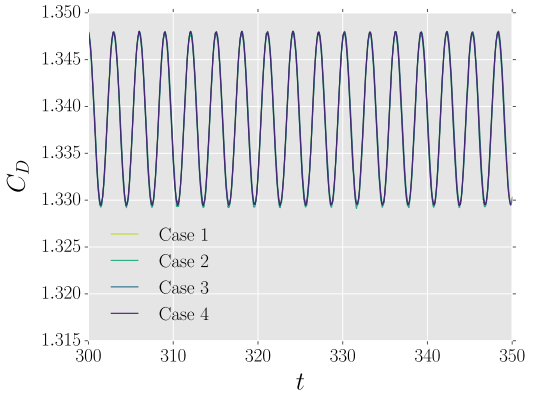

Fig. 3. Temporal evolution of the drag coefficient for the 2D cylinder case between $t = 300$ and $t = 350$. The graphs of Cases 2, 3 and 4 have been shifted in t such that they are in phase with Case 1. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

acceleration was performed with a 5-level cycle 1-1-1-1-2-1-1-1-3 corresponding to polynomial orders 4-3-2-1-0-1-2-3-4 and α_τ was set to 2.0 and 1.6, respectively. In Cases 2 and 4, the locally adaptive pseudo-time stepping parameters were set as $f_{\min} = 0.98$, $f_{\max} = 1.01$, $\Delta\tau_{min} = 4 \times 10^{-4}$ and $f_\tau = 8.0$. All cases were run with a convergence tolerance such that the point-wise L^2-norm of the largest velocity residual was always driven below 10^{-5}. The L^2-norm of the velocity divergence, which is directly proportional to pressure residual, was approximately 10^{-4} for each case.

6.2. Results

Table 3 shows the runtime parameters of all cases together with mean drag coefficients $\overline{C}_D$, Strouhal numbers St, and wall-times. The statistics were measured for a period of 100 time units in a fully developed quasi-steady-state from $t = 250.1250$ to $t = 350.1250$. It can be seen that all cases produce identical mean $\overline{C}_D$ and St, which are in line with values of Cox et al. ($\overline{C}_{Dref} = 1.338$ and $St_{ref} = 0.166$) who used implicit pseudo-time stepping with LU-SGS. Moreover, Fig. 3 shows that temporal evolution of the drag coefficients overlap. The graphs of Cases 2, 3 and 4 have been shifted in t such that they are in the same phase as Case 1 because of slight differences in initial transients. From the wall-times, we can see that locally adaptive pseudo-time stepping gives a speed-up factor of 4.16 relative to Case 1. Although not as effective as P-multigrid acceleration with a speed-up factor of 6.27, it is still substantial. Combining the two techniques together yields a total speed-up factor of 15.24 relative to Case 1.

6.3. Adaptivity analysis

To better understand the underlying adaptation mechanisms, temporal and spatial behaviour of local pseudo-time step sizes were studied with Case 4. Figs. 4a-d show the instantaneous z-vorticity field ω_z and fields of local pseudo-time step sizes $\Delta\tau_{en\alpha}$, with $\alpha \in \{p,\ v_x, v_y\}$, at $t = 340.35$. The pseudo-time step fields were averaged over solution points after the convergence criterion had been reached. Two observations can be made. First, the approach is able to adapt to the element element size and shape. This is the most evident in the structured boundary layer block and its immediate vicinity, where the pseudo-time step sizes grow monotonically with increasing element size. Second, the approach is able to adapt to the solution state, and vary between different field variables. Adaptation to solution state is most visible for $\Delta\tau_{enp}$, where areas of reduced pseudo-time step resemble the von Karman vortices in the wake section. The adaptation to different field variables can be seen in the boundary layer where $\Delta\tau_{enp}$ exhibits considerably larger values than $\Delta\tau_{env_x}$ and $\Delta\tau_{env_y}$. Fig. 5 shows the temporal evolution of the pseudo-time step sizes $\Delta\tau_\alpha$ for a solution point of level $l = P$ at $\mathcal{P} = (8.474,\ 1.155)$, which is in a triangular element located near the centre of the upper vortex street. The location of the probe is illustrated as a cross in Figs. 4a-d. It can be seen that all pseudo-time components experience a periodic reduction with a frequency that is half of the shedding frequency. This demonstrates that all pseudo-time step components at the probe location reduce and rise back up as a vortex passes.

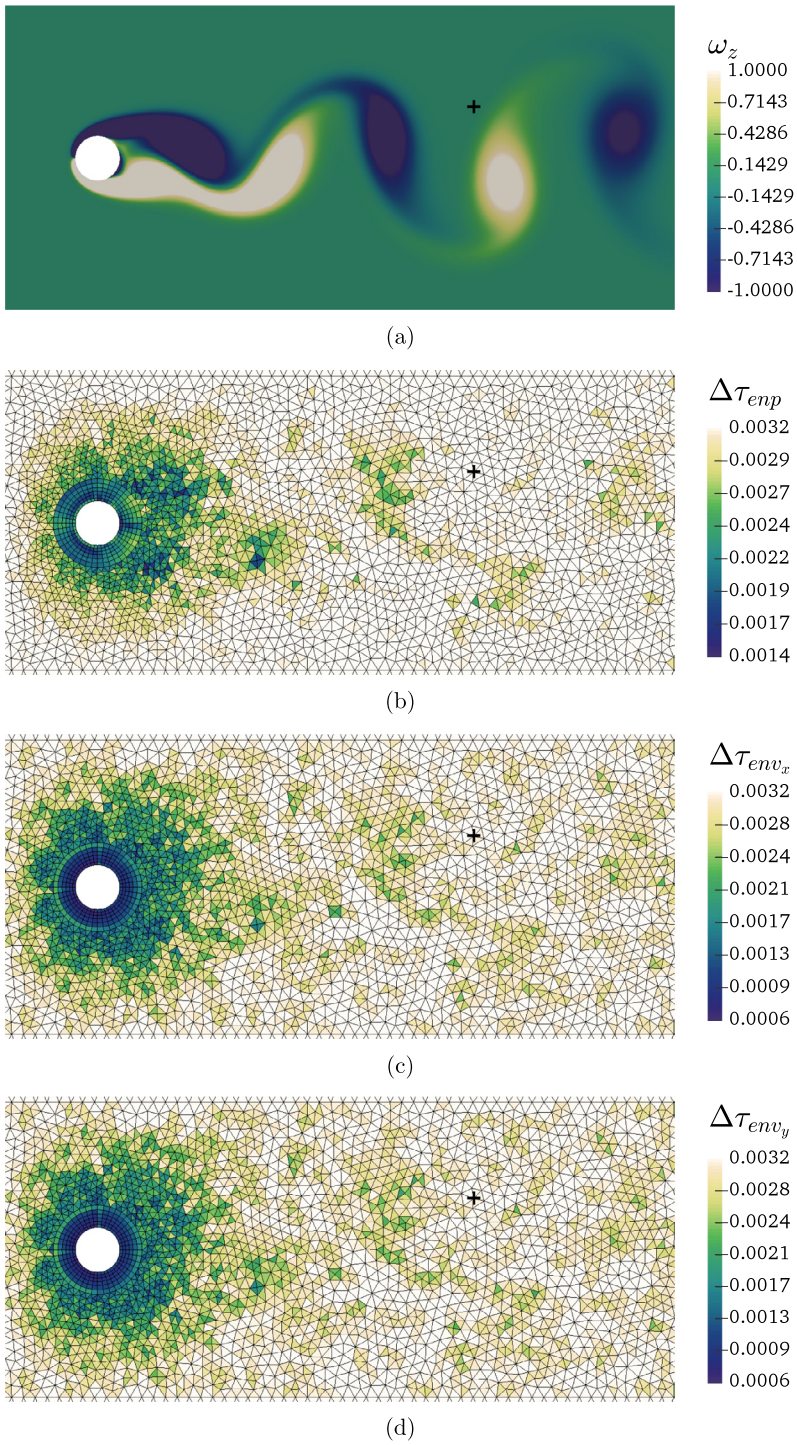

Fig. 4. (a) Instantaneous vorticity field ω_z at $t = 340.35$ for Case 4. Fields of pseudo-time step sizes (b) $\Delta\tau_{enp}$, (c) $\Delta\tau_{env_x}$ and (d) $\Delta\tau_{env_y}$ at $t = 340.35$ for Case 4. The cross is the location of the probe $\mathcal{P}$.

Figs. 6a-c show a close-up of the pseudo-time step sizes associated with individual solution points $\Delta\tau^u_{ejn\alpha}$ at level $l = P$ near the cylinder trailing edge at $t = 340.35$. It can be seen that the solution points closest to the element corners are the most restrictive, closely followed by the other solution points near the interfaces. The interior solution points are considerably less restrictive than the interface points. This observation supports our assertion that locally pseudo-time stepping approaches based on a reference length would be difficult to implement for nodal high-order schemes.

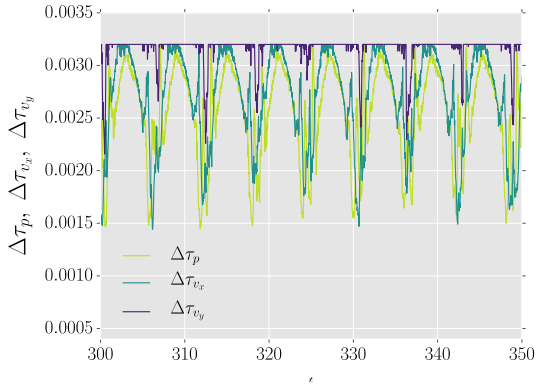

Fig. 5. Temporal evolution of $\Delta\tau_\alpha$ at probe $\mathcal{P}$, which is in a triangular element.

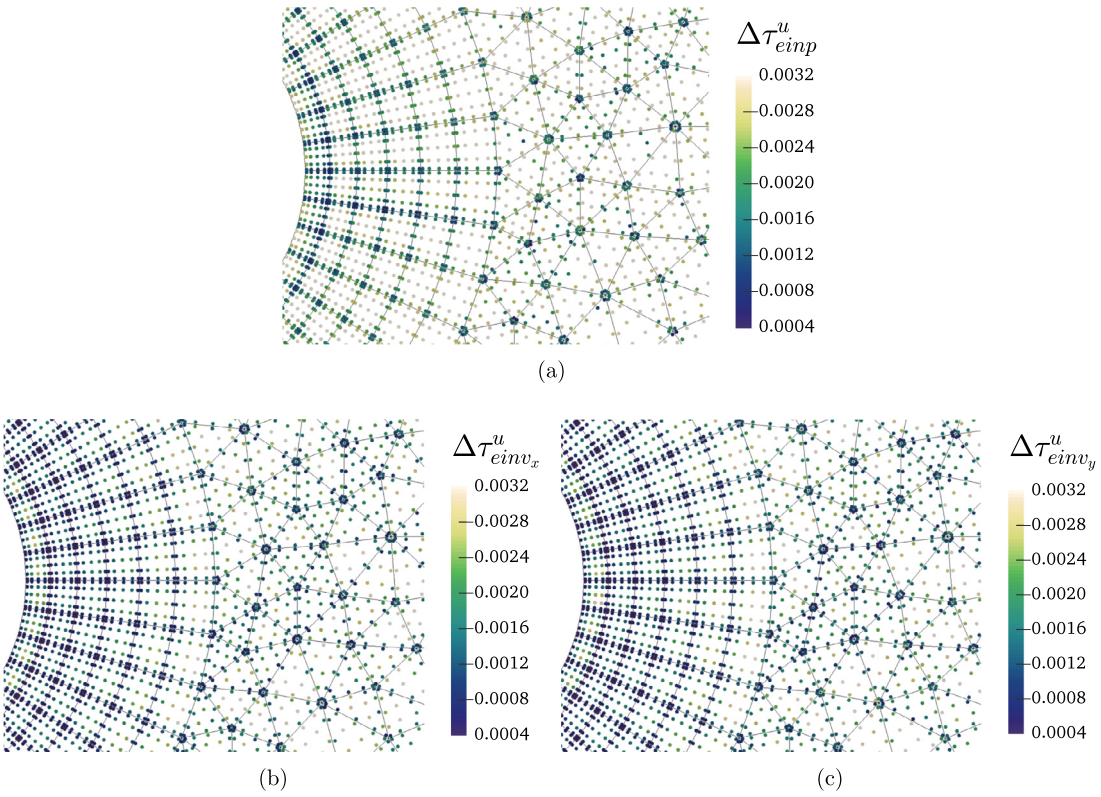

Fig. 6. Fields of pseudo-time step sizes (a) $\Delta\tau^u_{einp}$, (b) $\Delta\tau^u_{einv_x}$ and (c) $\Delta\tau^u_{einv_x}$ at individual solution points of $l = P$ near the cylinder trailing edge at $t = 340.35$ for Case 4.

6.4. Parameter sensitivity

Appropriate selection of $\Delta\tau_{\min}$, f_τ, and α_τ depends on the specifics of a particular simulation e.g. the solution polynomial order and the physical time-step size. In particular, varying α_τ and f_τ for Case 4 can have a large effect on the speed-up factor relative to Case 1, as shown in Fig. 7. The uncoloured regions represent parameter combinations that failed to converge. To contextualise, lower bounds for the parameters are $\alpha_\tau = 1$ and $f_\tau = 1$, which correspond to keeping the pseudo-time step sizes constant across all P-multigrid levels and keeping the pseudo-time step sizes constant across all solution points and field variables, respectively. It can be seen that significant speed-up factors can be achieved for a range of α_τ and f_τ, but finding the optimal values requires a degree of experimentation. The best performance is achieved by increasing both parameters together.

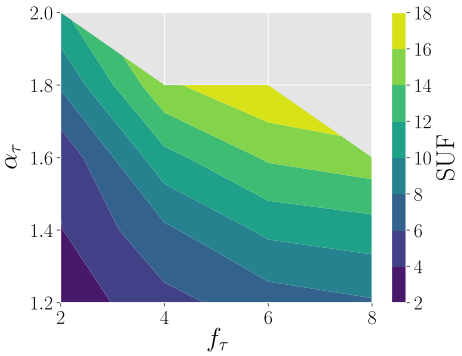

Fig. 7. The effect of varying f_τ and α_τ for Case 4 on the speed-up factor SUF relative to Case 1.

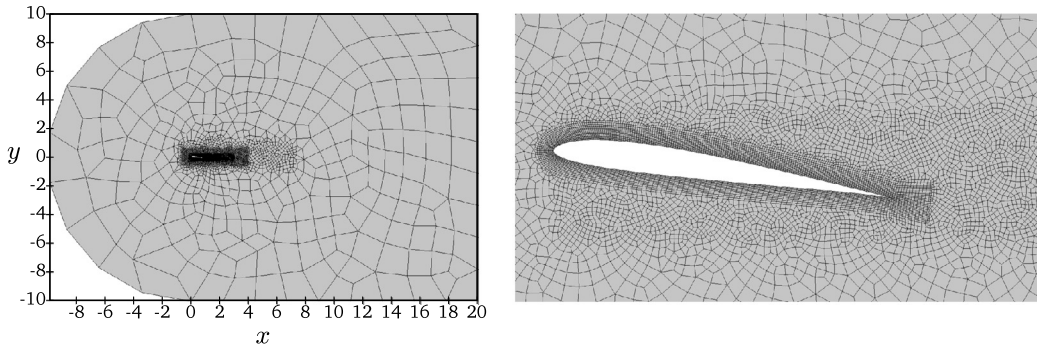

Fig. 8. The computational grid used in the 3D SD7003 airfoil simulations (left). A zoom close to the airfoil body (right).

7. SD7003 airfoil at $Re = 60,000$

7.1. Problem specification

An SD7003 airfoil simulation was performed at an 8° angle-of-attack and $Re = 60,000$, based on the chord length and free-stream velocity. Fig. 8 shows the unstructured computational grid used in the study. With the chord length being 1, the domain consists of an inflow section with a diameter of 20, and a rectangular wake section with a length of 20. The domain width in the span-wise direction was selected as 0.2. The total element count was 137,916, consisting of only hexahedra. A Riemann-invariant-based boundary condition, detailed in Appendix A, was prescribed at all far-field boundaries with $u_\infty = \{1\ 1\ 0\ 0\}$ and $\zeta_\infty = 300$. Also in this case the boundary condition was found to be substantially less reflective than conventional BCs in [5,6].

No-slip condition was prescribed on the airfoil surface. The initial condition at $t = 0$ was set as $u = \{1\ 1\ 0\ 0\}$. The simulations were performed with a patched version of PyFR v1.8.0. The patch, configuration files and the mesh are provided as Electronic Supplementary Material.

One complete simulation with $P = 4$ and $\zeta = 3$ was on run 32 Nvidia P100 GPUs using double precision. The physical time was discretised with BDF2, where $\Delta t = 1 \times 10^{-3}$. Locally adaptive pseudo-time stepping with P-multigrid convergence acceleration was used in pseudo-time with $f_{\text{min}} = 0.998$, $f_{\text{max}} = 1.001$, $\Delta\tau_{min} = 3.6 \times 10^{-5}$, and $f_\tau = 7.0$. The P-multigrid cycle was prescribed as 1-1-1-1-4-1-1-1-6, corresponding to polynomial orders 4-3-2-1-0-1-2-3-4, with $\alpha_\tau = 1.7$. Additionally, flux anti-aliasing with quadrature degrees 11-9-7-5-3-5-7-9-11 was performed at each P-multigrid level. The simulation was run in three stages. First, the flow was developed with $P = 1$ up to $t = 25.011$. Second, the simulation was restarted and run up to $t = 45.024$ with $P = 4$ to settle transients. Third, flow statistics were measured between $t = 45.024$ and $t = 70.043$. The case was run with a convergence tolerance such that the point-wise L^2-norm of the largest velocity residual was always driven below 10^{-4}. The L^2-norm of the velocity divergence, which is directly proportional to pressure residual, was approximately 10^{-3}.

7.2. Results

Fig. 9 plots Q-criterion iso-surfaces coloured by instantaneous velocity V at $t = 59.5460$. The simulation is capable of capturing the characteristics of the test case as described in [28]; a laminar separation bubble is formed at the leading edge which breaks up into turbulence via Kelvin-Helmholtz vortices. Qualitatively the turbulent structures are well-resolved throughout the domain. Table 4 shows a comparison of the simulation data against compressible low-Mach data

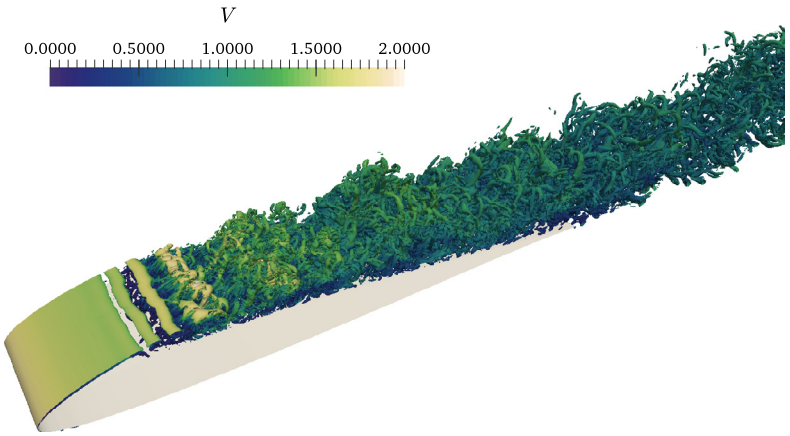

Fig. 9. Q-criterion iso-surfaces coloured by instantaneous velocity V at t=59.546.

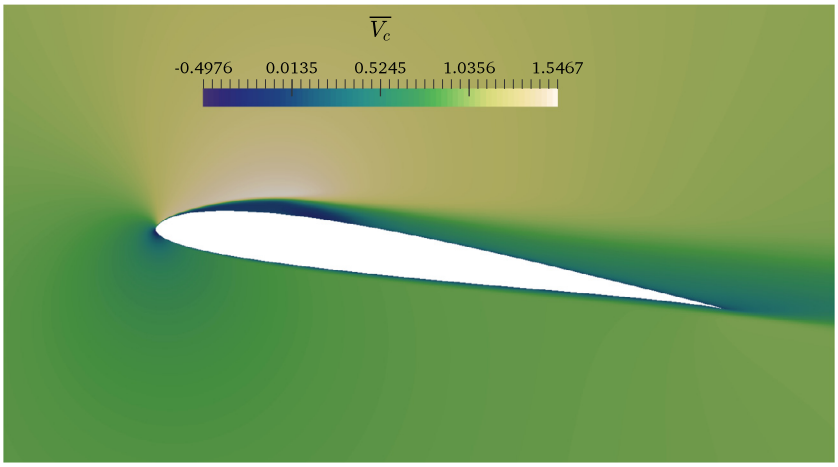

Fig. 10. Time and span-averaged velocity field in the chord-wise direction $\overline{V_c}$.

Table 4
The current simulation data and reference data for the SD7003 case at Re=60,000. INS abbreviates Incompressible Navier-Stokes and NS the compressible Navier-Stokes equations.

	Formulation	Method	$\overline{C}_D$	$\overline{C}_L$	x_{sep}	x_{rea}
Current	AC INS	FR $P=4$	0.052	0.923	0.033	0.346
Beck et al. [28]	$Ma=0.1$ NS	DG $P=8$	0.050	0.932	0.030	0.336
Beck et al. [28]	$Ma=0.1$ NS	DG $P=3$	0.045	0.923	0.027	0.310
Vermeire et al. [29]	$Ma=0.2$ NS	FR $P=4$	0.059	0.941	0.045	0.315
Selig et al. [30]	–	Exp.	0.029	0.920	–	–

by Beck et al. [28] using DG, Vermeire et al [29] using FR, and low-speed experiments by Selig et al. [30]. The separation and reattachement points were measured from the time and span-averaged velocity in the chord-wise direction $\overline{V}_c = v_x\cos(8^\circ) + v_y\sin(8^\circ)$, which is visualised in Fig. 10. All numerical results appear to over-estimate the drag coefficient C_D compared to the experiments. The biggest differences across the numerical results are observed in the lift coefficients $\overline{C}_L$ and reattachement points x_{rea}. The current simulation results are within the bounds of those reported by [28,29] except for x_{rea} which was found to be higher, but still close to the high-resolution $P=8$ results of Beck et al.

In addition to the complete simulation, four additional runs starting from $t=45.532$ were undertaken for a single flow pass over the chord. Case 1 was performed without any convergence acceleration using global RK4 pseudo-time stepping with a constant $\Delta\tau = 3.7\times10^{-5}$ that was optimised with a bisection approach. Case 2 was performed with locally adaptive RK3(2)4[2R+] pseudo-time stepping alone, Case 3 with P-multigrid acceleration alone, and Case 4 with the combination of the two. In Cases 3 and 4, P-multigrid convergence acceleration was performed with a 5-level cycle 1-1-1-1-4-1-1-1-6 corresponding to polynomial orders 4-3-2-1-0-1-2-3-4 and α_τ was set to 1.7. In Cases 2 and 4, the locally adaptive pseudo-time stepping parameters were set as $f_{\min}=0.998$, $f_{\max}=1.001$, $\Delta\tau_{min}=3.6\times10^{-5}$ and $f_\tau=7.0$. All cases were run with a convergence tolerance such that the point-wise L^2-norm of the largest velocity residual was always driven below 10^{-4}. The

Table 5
Summary of the SD7003 performance runs, where PS denotes the pseudo-stepper, LAPTS denotes locally adaptive pseudo-time stepping, P-MG denotes P-multigrid, WT denotes the wall-time, and SUF denotes the speed-up factor relative to Case 1.

	PS	LAPTS	P-MG	f_τ	α_τ	WT	SUF
Case 1	RK4	Off	Off	–	–	21:43:08	1.00
Case 2	RK3(2)4[2R+]	On	Off	7	–	08:50:47	2.46
Case 3	RK4	Off	On	–	1.7	04:28:58	4.84
Case 4	RK3(2)4[2R+]	On	On	7	1.7	01:51:54	11.65

L^2-norm of the velocity divergence, which is directly proportional to pressure residual, was approximately 10^{-3} for each case. Table 5 shows the runtime parameters, wall-times, and speed-up factors relative to Case 1. A speed-up factor of 2.46 is achieved by locally adaptive pseudo-time stepping alone. The speed-up factor with the combination of locally adaptive pseudo-time stepping and P-multigrid acceleration was 11.65.

8. Conclusion

A novel locally adaptive pseudo-time stepping convergence acceleration technique for dual time stepping has been proposed. In contrast to standard local pseudo-time stepping techniques that are based on computing the local pseudo-time steps directly from estimates of the local CFL limit, the proposed technique controls the local pseudo-time steps using local truncation errors which are computed with embedded pair RK schemes. The approach has three advantages. First, it does not require an expression for the characteristic element size, which are difficult to obtain reliably for curved mixed-element meshes. Second, it allows a finer level of locality for high-order nodal discretisations, such as FR, since the local time-steps can vary between solution points and field variables. Third, it is well-suited to being combined with P-multigrid convergence acceleration. Results were presented for a laminar 2D cylinder test case at $Re = 100$. A speed-up factor of 4.16 was achieved compared to global pseudo-time stepping with and RK4 scheme, while maintaining accuracy. Moreover, when combined with P-multigrid convergence acceleration a speed-up factor of over 15 was achieved. Detailed analysis of the results reveals that pseudo-time steps adapt to element size/shape, solution state, and solution point location within each element. Finally, results were presented for a turbulent 3D SD7003 airfoil test case at $Re = 60,000$. A speed-up factor of 2.46 was achieved compared to global pseudo-time stepping with an RK4 scheme, and the flow physics was found to be in good agreement with previous studies. Moreover, when combined with P-multigrid convergence acceleration a speed-up factor of over 11 was achieved.

Future work should focus on three areas. The first should look to extend the approach for use with higher-order physical time stepping methods such as diagonally implicit Runge-Kutta schemes, which can be beneficial for applications that involve very small and/or rapidly varying time-scales [31]. The second should look to develop ways of aiding or automating parameter selection, in particular α_τ and f_τ, for specific test cases. Finally, future work should look at the effect of using different explicit schemes for pseudo-time stepping, including those with limited time-accuracy such as simple forwards Euler schemes, or embedded pair extensions of stability optimised multi-stage schemes, such as those recently developed in [32].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

N.A. Loppi and P.E. Vincent would like to thank the Engineering and Physical Sciences Research Council and BAE Systems for their support via an Industrial CASE doctoral training grant and an Early Career Fellowship (EP/K027379/1, EP/R030340/1). F.D. Witherden and A. Jameson would like to thank the Air Force Office of Scientific Research for their support via grant FA9550-14-1-018. This work was also supported by compute allocations on Piz Daint at the Swiss National Supercomputing Centre under project ID S833 and Peta4-KNL at the Cambridge Service for Data Driven Discovery under project ID CS005.

Appendix A. Characteristic Riemann-invariant boundary condition for ACM

In PyFR, boundary conditions are imposed via ghost states. When computing the common flux for flux points at the domain boundary with $\mathcal{F}(u_\mathrm{L}, \nabla \mathbf{u}_\mathrm{L}, u_\mathrm{R}, \nabla \mathbf{u}_\mathrm{R}, \hat{\mathbf{n}})$, the right-hand-side solution state u_R is replaced with a prescribed ghost state $\mathcal{B}^{\mathrm{Rie}} u_\mathrm{R}$ and the right-hand-side gradient state $\nabla \mathbf{u}_\mathrm{R}$ is replaced with a gradient ghost state $\mathcal{B}^{\mathrm{LDG}} \nabla \mathbf{u}_\mathrm{R}$. Furthermore, the right-hand-side solution state in $\mathcal{C}(u_\mathrm{R}, u_\mathrm{L})$ is replaced with the viscous ghost state $\mathcal{B}^{\mathrm{LDG}} u_\mathrm{R}$ which can differ from $\mathcal{B}^{\mathrm{Rie}} u_\mathrm{R}$.

The characteristic boundary condition used in the current study is based on Riemann invariants derived in [4]. The boundary condition is parametrised by a free-stream pressure p_∞, free-stream velocity $\mathbf{v}_\infty$ and free-stream artificial compressibility factor ζ_∞. It was found empirically that setting ζ_∞ to be larger than the global artificial compressibility factor ζ made the boundary less reflective. The 1D Riemann invariants in the interface normal direction are defined as

$$\Gamma_\infty = p_\infty + \frac{1}{2}\left[v_\infty^\perp(v_\infty^\perp - c_\infty) - \zeta_\infty \log(v_\infty^\perp + c_\infty)\right], \tag{A.1}$$

$$\Gamma_L = p_\mathrm{L} + \frac{1}{2}\left[v_\mathrm{L}^\perp(v_\mathrm{L}^\perp + c_\mathrm{L}) + \zeta_\infty \log(v_\mathrm{L}^\perp + c_\mathrm{L})\right], \tag{A.2}$$

where p_L is the pressure interpolated from the interior solution, $v_\mathrm{L}^\perp = \hat{\mathbf{n}} \cdot \mathbf{v}_L$, with $\mathbf{v}_L$ being the velocity interpolated from the interior domain, $v_\infty^\perp = \hat{\mathbf{n}} \cdot \mathbf{v}_\infty$, $c_\mathrm{L} = \sqrt{(v_\mathrm{L}^\perp)^2 + \zeta_\infty}$, and $c_\infty = \sqrt{(v_\infty^\perp)^2 + \zeta_\infty}$. Using these relations, the normal velocity at the boundary can be computed using a Newton-Raphson method as

$$v_b^{\perp,i+1} = v_b^{\perp,i} - \frac{f(v_b^{\perp,i})}{f'(v_b^{\perp,i})}, \tag{A.3}$$

where the superscript i is the iteration counter, and

$$f(v_b^\perp) = c_b v_b^\perp + \zeta_\infty \log(v_b^\perp + c_b) + \Gamma_\infty - \Gamma_\mathrm{L}, \tag{A.4}$$

$$f'(v_b^\perp) = 2\frac{(v_b^\perp)^2 + \zeta_\infty}{c_b}, \tag{A.5}$$

with $c_b = \sqrt{(v_b^\perp)^2 + \zeta_\infty}$. The process converges to machine precision in less than 5 iterations which in PyFR v1.8.0 are fully unrolled by the templating engine. Using the converged $v_b^\perp$, the individual velocity components at the boundaries can be calculated as

$$\mathbf{v}_b = \begin{Bmatrix} \mathbf{v}_\mathrm{L} + \hat{\mathbf{n}}(v_b^\perp - \hat{\mathbf{n}} \cdot \mathbf{v}_\mathrm{L}) & \text{if } \hat{\mathbf{n}} \cdot \mathbf{v}_\mathrm{L} \geq 0 \\ \mathbf{v}_\infty + \hat{\mathbf{n}}(v_b^\perp - \hat{\mathbf{n}} \cdot \mathbf{v}_\infty) & \text{if } \hat{\mathbf{n}} \cdot \mathbf{v}_\mathrm{L} < 0 \end{Bmatrix}. \tag{A.6}$$

Finally, the boundary ghost states are prescribed as

$$\mathcal{B}^{\mathrm{Rie}} u_\mathrm{R} = \mathcal{B}^{\mathrm{LDG}} u_\mathrm{R} = \begin{Bmatrix} \Gamma_\mathrm{L} - \frac{1}{2}\left[v_b(v_b^\perp + c_b + \zeta_\infty \log(v_b^\perp + c_b)\right] \\ \mathbf{v}_b \end{Bmatrix}, \tag{A.7}$$

$$\mathcal{B}^{\mathrm{LDG}} \nabla \mathbf{u}_\mathrm{R} = 0. \tag{A.8}$$

Appendix B. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jcp.2019.108913.

References

[1] A. Jameson, Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings, in: AIAA 10th Comput. Fluid Dyn. Conf., 1991, pp. 1–14.
[2] E. Turkel, V.N. Vatsa, Choice of variables and preconditioning for time dependent problems eli turkel, in: AIAA 16th Comput. Fluid Dyn. Conf., 2003, pp. 1–9.
[3] A.J. Chorin, A numerical methods for solving incompressible viscous flow problems, J. Comput. Phys. 135 (1997) 118–125.
[4] F. Bassi, A. Crivellini, D.A. Di Pietro, S. Rebay, An artificial compressibility flux for the discontinuous Galerkin solution of the incompressible Navier-Stokes equations, J. Comput. Phys. 218 (2) (2006) 794–815.
[5] C. Liang, A. Chan, X. Liu, A. Jameson, An artificial compressibility method for the spectral difference solution of unsteady incompressible Navier-Stokes equations on multiple grids, in: 49th AIAA Aerosp. Sci. Meet. Incl. New Horizons Forum Aerosp. Expo., 2011, pp. 1–13.
[6] C. Cox, C. Liang, M.W. Plesniak, A high-order solver for unsteady incompressible Navier-Stokes equations using the flux reconstruction method on unstructured grids with implicit dual time stepping, J. Comput. Phys. 314 (2016) 414–435.
[7] N.A. Loppi, F.D. Witherden, A. Jameson, P.E. Vincent, A high-order cross-platform incompressible Navier-Stokes solver via artificial compressibility with application to a turbulent jet, Comput. Phys. Commun. 233 (2018) 193–205.
[8] S. Minisini, E. Zhebel, A. Kononov, W. Mulder, Local time stepping with the discontinuous Galerkin method for wave propagation in 3D heterogeneous media, Geophysics 78 (2013) 67–77.
[9] P. Nithiarasu, An efficient artificial compressibility (AC) scheme based on the characteristic based split (CBS) method for incompressible flows, Int. J. Numer. Methods Eng. 56 (13) (2003) 1815–1845.
[10] H.T. Huynh, A flux reconstruction approach to high-order schemes including discontinuous Galerkin methods, in: 18th AIAA Comp. Fluid Dyn. Conf., 2007, pp. 1–42.
[11] H.T. Huynh, A reconstruction approach to high-order schemes including discontinuous Galerkin for diffusion, in: 47th AIAA Aerosp. Sci. Meet. Incl. New Horizons Forum Aerosp. Expo., 2009, pp. 1–34.

[12] P.E. Vincent, P. Castonguay, A. Jameson, A new class of high-order energy stable flux reconstruction schemes, J. Sci. Comput. 47 (1) (2011) 50–72.
[13] P.E. Vincent, P. Castonguay, A. Jameson, Insights from von Neumann analysis of high-order flux reconstruction schemes, J. Comput. Phys. 230 (22) (2011) 8134–8154.
[14] A. Jameson, P.E. Vincent, P. Castonguay, On the non-linear stability of flux reconstruction schemes, J. Sci. Comput. 50 (2) (2012) 434–445.
[15] F. Witherden, P. Vincent, An analysis of solution point coordinates for flux reconstruction schemes on triangular elements, J. Sci. Comput. (2014) 398–423.
[16] L. Shunn, F. Ham, Symmetric quadrature rules for tetrahedra based on a cubic close-packed lattice arrangement, J. Comput. Appl. Math. 236 (17) (2012) 4348–4364.
[17] J.S. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications, Springer Science & Business Media, 2007.
[18] E. Hairer, H. Wanner, Solving Ordinary Differential Equations II - Stiff and Differential-Algebraic Problems, 2nd edition, Springer, 1996.
[19] C.A. Kennedy, M.H. Carpenter, R.M. Lewis, Low-storage, explicit Runge-Kutta schemes for the compressible Navier-Stokes equations, Appl. Numer. Math. 35 (3) (2000) 177–219.
[20] Z. Yang, D. Mavriplis, Unstructured dynamic meshes with higher-order time integration schemes for the unsteady Navier-Stokes equations, in: 43rd AIAA Aerosp. Sci. Meet. Exhib., 2005, pp. 1–13.
[21] D.T. Elsworth, E.F. Toro, A Numerical Investigation of the Artificial Compressibility Method for the Solution of the Navier-Stokes Equations, Tech. Rep. 9213, Cranfield Institute of Technology, 1992.
[22] K.J. Fidkowski, T. Oliver, J. Lu, D.L. Darmofal, p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations, J. Comput. Phys. 207 (1) (2005) 92–113.
[23] A. Jameson, Transonic Flow Calculations, Von Karman Inst. for Fluid Dynamics Lecture Series, vol. 87, 1976, pp. 1–93.
[24] A.G. Malan, R.W. Lewis, P. Nithiarasu, An improved unsteady, unstructured, artificial compressibility, finite volume scheme for viscous incompressible flows: Part I. Theory and implementation, Int. J. Numer. Methods Eng. 54 (5) (2002) 695–714.
[25] C.G. Thomas, P. Nithiarasu, Influences of element size and variable smoothing on inviscid compressible flow solution, Int. J. Numer. Methods Heat Fluid Flow 15 (5) (2005) 420–428.
[26] E. Jones, T. Oliphant, P. Peterson, et al., SciPy: Open source scientific tools for Python, [Online; accessed 2019-07-08] (2001–). http://www.scipy.org/.
[27] B. Houska, H.J. Ferreau, M. Diehl, ACADO toolkit - an open-source framework for automatic control and dynamic optimization, Optim. Control Appl. Methods 32 (2011) 298–312.
[28] A.D. Beck, T. Bolemann, D. Flad, H. Frank, G.J. Gassner, F. Hindenlang, C.D. Munz, High-order discontinuous Galerkin spectral element methods for transitional and turbulent flow simulations, Int. J. Numer. Methods Fluids 76 (8) (2014) 522–548.
[29] B.C. Vermeire, F.D. Witherden, P.E. Vincent, On the utility of GPU accelerated high-order methods for unsteady flow simulations: a comparison with industry-standard tools, J. Comput. Phys. 334 (2017) 497–521.
[30] M.S. Selig, B.D. McGranahan, Summary of Low speed Airfoil Data, vol. 1, 1995.
[31] O. Peles, E. Turkel, Adaptive time steps for compressible flows based on dual-time stepping and a RK/implicit smoother scheme, in: Tenth Int. Conf. Comput. Fluid Dyn. ICCFD10, 2018, pp. 1–14.
[32] B.C. Vermeire, N.A. Loppi, P.E. Vincent, Optimal Runge-Kutta schemes for pseudo time-stepping with high-order unstructured methods, J. Comput. Phys. 383 (2019) 55–71.