Journal of Computational Physics 229 (2010) 5944–5965

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

An adaptive implicit–explicit scheme for the DNS and LES of compressible flows on unstructured grids

Mohammad Shoeybi [a,*], Magnus Svärd [b], Frank E. Ham [a], Parviz Moin [a]

[a] Center for Turbulence Research, Stanford University, Stanford, CA 94305, USA
[b] School of Mathematics, University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh, EH9 3JZ, United Kingdom

ARTICLE INFO

Article history:
Received 24 August 2009
Received in revised form 12 April 2010
Accepted 14 April 2010
Available online 29 April 2010

Keywords:
Compressible flows
DNS
LES
Unstructured grids
Time-advancement scheme
IMEX
SBP
SAT

ABSTRACT

An adaptive implicit–explicit scheme for Direct Numerical Simulation (DNS) and Large-Eddy Simulation (LES) of compressible turbulent flows on unstructured grids is developed. The method uses a node-based finite-volume discretization with Summation-by-Parts (SBP) property, which, in conjunction with Simultaneous Approximation Terms (SAT) for imposing boundary conditions, leads to a linearly stable semi-discrete scheme. The solution is marched in time using an Implicit–Explicit Runge–Kutta (IMEX-RK) time-advancement scheme. A novel adaptive algorithm for splitting the system into implicit and explicit sets is developed. The method is validated using several canonical laminar and turbulent flows. Load balance for the new scheme is achieved by a dual-constraint, domain decomposition algorithm. The scalability and computational efficiency of the method is investigated, and memory savings compared with a fully implicit method is demonstrated. A notable reduction of computational costs compared to both fully implicit and fully explicit schemes is observed.

1. Introduction

Direct Numerical Simulation (DNS) and Large-Eddy Simulation (LES) are widely used to simulate compressible turbulent flows. DNS simulates all the flow scales, whereas in LES, the equations are low-pass filtered, and the small-scale turbulent eddies are modeled. Recently, there has been a growing interest in applying DNS and especially LES to practical engineering applications that typically involve high Reynolds number flows and complex geometries. An unstructured spatial discretization scheme is the preferred choice to resolve complex geometries. Both the computational grid and the flow may induce stiffness in the equations restricting the time-step size of an explicit time integrator. Typical examples include resolved boundary layers for which both the viscous and inviscid terms may be stiff. The former is due to the small grid size and the latter is due to the acoustic Courant–Friedrichs–Lewy (CFL) condition.

Several approaches have addressed stiff equations. A fully implicit approach treats every term throughout the entire computational domain implicitly. However, it requires the solution of a large non-linear system of equations. For realistic applications, the cost of solving the non-linear system may be more than marching the scheme explicitly in time. Some methods, such as relaxation-based schemes and multi-grid reduce the cost of a fully implicit approach (see [1], and for unstructured methods [2] and references therein). Furthermore, the memory required to store the Jacobian matrix and preconditioners is considerable and may prevent computations of realistic applications. The memory requirement for the Jacobian matrix can be reduced by means of a pressure-correction method [3,4] or eliminated by using a Jacobian-free Krylov subspace linear

* Corresponding author. Tel.: +1 650 723 2416.
E-mail address: shoeybi@stanford.edu (M. Shoeybi).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.04.027

solver [5]. For the latter method, the sparse matrix–vector multiplication is replaced by a flux calculation, which increases the computational cost. Hence, there is considerable room for designing effective numerical schemes by avoiding fully implicit time-integration schemes.

One approach to reduce both memory and computational cost is to identify stiff parts of the Ordinary Differential Equation (ODE) system obtained from spatial discretization of the governing PDE and employ an implicit–explicit (IMEX) time-integration scheme. An IMEX scheme integrates stiff parts implicitly and the non-stiff parts explicitly in time. Such a time-advancement scheme can be constructed using linear multi-step methods [6]. Alternatively, one may use Runge–Kutta based IMEX schemes [7,8] that have better stability properties than linear multi-step methods. In any case, an IMEX method requires partitioning of the ODE system into stiff and non-stiff parts. The stiffness may originate from specific terms in the governing equations (see for instance, [9–11]) and/or from specific directions/regions in the flow (for example, the wall-normal diffusion in a boundary-layer simulation [12]). However, for the unstructured grids it is difficult to identify a stiff direction. For specially designed unstructured grids, Nompelis et al. [13] proposed a method with an implicit treatment of the wall-normal direction. Nevertheless, in many cases, the stiffness is confined to certain regions in the computational domain. A possible approach for such problems is to use a variable hybrid method that blends explicit and implicit schemes using a continuous parameter controlling the fraction of each problem [14]. Alternatively, one can decompose the computational domain into implicit and explicit regions using a measure for geometrically induced stiffness (see Kanevsky et al. [15]).

Therefore, there is potential for saving computational cost by designing an efficient algorithm that treats implicitly only those regions and/or phenomena that require it. In this paper, we propose a novel splitting algorithm that measures the stiffness and devise implicit and explicit parts, with respect to both the computational domain and the spatial directions. The splitting algorithm will be referred to as Row-Splitted IMEX (RS-IMEX) scheme. The RS-IMEX scheme operates in time allowing the decomposition to adapt at every time step. Moreover, the scheme is well-suited for large-scale computations required for the DNS/LES, as it is designed to be highly parallelizable and scalable.

In this study, we use a node-based finite-volume scheme to discretize the spatial derivatives. In [16–18], the SBP and SAT techniques for imposing boundary conditions were used to prove stability for high-order finite difference approximations of the Navier–Stokes equations. Furthermore, the SBP property was proved for node-based unstructured finite-volume schemes in [19–21]. In Section 2, we utilize these results to obtain a stable finite-volume discretization. In Section 3, we develop the RS-IMEX scheme for the advection–diffusion equation and generalize it to the Navier–Stokes equations on unstructured grids. Finally, we validate the proposed scheme and demonstrate its performance using several test cases in Section 4.

2. Governing equations and spatial discretization

The governing equations are the compressible Navier–Stokes, energy and continuity equations. Let Ω and $\partial\Omega$ be the computational domain and its boundary. By using L_r, ρ_r, u_r, T_r, and μ_r as reference length, density, velocity, temperature, and molecular viscosity, the equations can be stated as

$$\frac{\partial}{\partial t}(\underset{\sim}{\vec{U}}) + \sum_{j=1}^{3} \frac{\partial}{\partial x_j}\left(\vec{F}_j - \frac{1}{Re}\vec{G}_j\right) = 0, \tag{1}$$

where $\underset{\sim}{\vec{U}}^T = (\rho, \rho\vec{u}^T, \rho E)^T$ denotes the conservative variables and

$$\vec{F}_j = \begin{pmatrix} \rho u_j \\ \rho u_i u_j + p\delta_{ij} \\ (\rho E + p)u_j \end{pmatrix}, \quad \vec{G}_j = \begin{pmatrix} 0 \\ \tau_{ij} \\ u_i\tau_{ij} + q_j \end{pmatrix}, \quad i = 1,2,3,$$

where ρ, p, u_i, E, and $Re = \rho_r u_r L_r/\mu_r$ denote density, pressure, velocities, energy, and Reynolds number, respectively. Furthermore,

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right), \quad q_j = \frac{\mu}{(\gamma-1)PrM_r^2}\frac{\partial T}{\partial x_j},$$

where $\mu = T^n$, $n = constant$, δ_{ij} is the Kronecker's delta, $M_r = \frac{u_r}{c_r}$, c_r is the reference sound speed, and Pr is the Prandtl number. The equations are supplemented with the state equation for an ideal gas, $p = \frac{\rho T}{\gamma M_r^2}$, and a suitable boundary operator, $L_{\partial\Omega} = g(\vec{x}, t)$ on $\partial\Omega$.

The domain Ω is discretized by an unstructured grid with M grid cells $\overline{\Omega}_c$ such that $\Omega = \bigcup_{c=1}^{M}\overline{\Omega}_c$. Eq. (1) is approximated by a finite-volume method on the dual volumes, Ω_k. In 2D, the dual volumes Ω_k are constructed by connecting the edge centers (ec), to the cell centers (cc), as depicted in Fig. 1. In 3D, the dual volumes are formed by the union of several triangular faces. Each face is constructed by connecting the edge center, the cell center, and the face center of the grid cells.

A finite-volume method defined by the dual volumes is usually referred to as a node-based scheme or a cell-vertex scheme. On a fixed dual volume Ω_k with fixed boundaries $\partial\Omega_k$ and outward-facing normal $\vec{n}$, the integral form of (1) is

$$\frac{\partial}{\partial t}\int_{\Omega_k} \underset{\sim}{\vec{U}}\, dV + \int_{\partial\Omega_k} \sum_{j=1}^{3}\left(\vec{F}_j - \frac{1}{Re}\vec{G}_j\right) dn_j = 0, \tag{2}$$

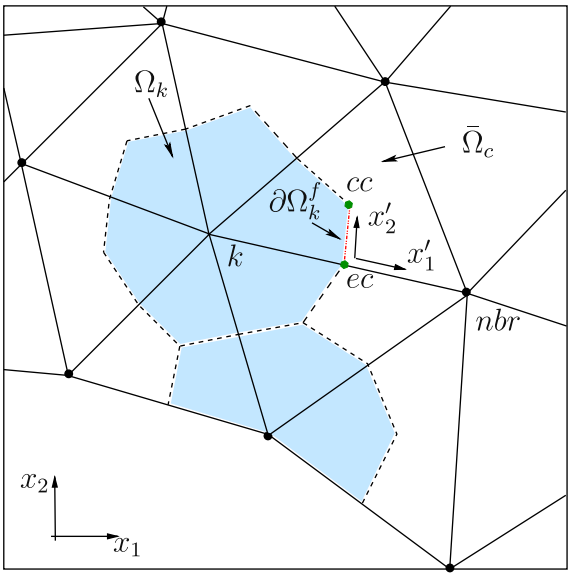

Fig. 1. Schematic of an unstructured grid with the dual volumes Ω_k.

where n_j is the jth component of the surface normal $\vec{n}$ multiplied by the surface area. As seen in Fig. 1, $\partial\Omega_k$ consists of a number of sub-boundary faces, $\partial\Omega_k^f$, that satisfy $\partial\Omega_k = \bigcup_f \partial\Omega_k^f$. We denote the components of the face normals by n_j^f. A node-based discretization of (2) can be written as

$$\frac{\partial \vec{U}_k}{\partial t} + \frac{1}{V_{\Omega_k}} \sum_f \sum_{j=1}^{3} \left(\vec{F}_j^f - \frac{1}{Re} \vec{G}_j^f \right) n_j^f = 0, \tag{3}$$

where V_{Ω_k} is the volume measure of Ω_k, and $\vec{U}_k$ are the unknown state variables at grid point k. $\vec{F}^f$ and $\vec{G}^f$ are approximations of the fluxes at the sub-boundary faces $\partial\Omega_k^f$, yet to be defined.

Let the superscript a represent the averaging between the node k and its neighbor nbr, e.g. $\rho^a = \frac{\rho^k + \rho^{nbr}}{2}$. We use a skew-symmetric form (see [22–25]) and define

$$\vec{F}_j^f = \begin{cases} (\rho^a u_j^a, (\rho u_i)^a u_j^a + p^a \delta_{ij}, (\rho E + p)^a u_j^a)^T, & \partial\Omega_k^f \notin \partial\Omega \\ \vec{F}_j^k & \partial\Omega_k^f \in \partial\Omega. \end{cases} \tag{4}$$

We also note that the difference between the central scheme obtained from simple averaging [19] and the skew-symmetric is a small built-in diffusion term (see [26]). Hence, the stability of the skew-symmetric scheme follows from the SBP–SAT properties.

Next, we define the viscous fluxes $\vec{G}^f$ on the subfaces $\partial\Omega_k^f$. We follow the same approximation used in Ham et al. [27]. For any function ϕ, the derivatives in 2D along the two independent directions x_1' and x_2' (see Fig. 1) are approximated as

$$\vec{\nabla}\phi|^f = \frac{\phi^{nbr} - \phi^k}{l^{k,nbr}} \hat{e}_1' + \frac{\phi^{cc} - \phi^{ec}}{l^{cc,ec}} \hat{e}_2', \tag{5}$$

where $\vec{\nabla}\phi|^f$ is the gradient approximation along the subface $\partial\Omega_k^f$. $\hat{e}_1'$ and $\hat{e}_2'$ are the unit vectors in x_1' and x_2' directions. $l^{k,nbr}$ is the distance between the nodes k and its neighbor nbr, and $l^{cc,ce}$ the distance between the cell center cc and edge center ec in the prime grid. Using this approximation for gradients on each subface, viscous fluxes $\vec{G}^f$ can be calculated.

Remark. On structured grids, this method of calculating viscous fluxes will reduce to the same discretization used in [28], which was shown to be an SBP operator. Ham et al. [27] studied this method for the Laplacian operator and showed that for simplex elements, the operator has the SBP property. They also demonstrated its stability for general polyhedral grid by performing numerical examples. To the best of the authors' knowledge, there is no stability proof for general unstructured grids, but the numerical examples, as well as the analysis of the resulting matrices, show that in absence of extreme element deformations, they are negative semi-definite and, therefore, stable.

We also use the far-field boundary conditions for SBP–SAT schemes proposed by Svärd et al. [18]. The far-field conditions specify data for a sum of the in-going characteristics and the viscous fluxes. Furthermore, we use the no-slip wall boundary conditions in Svärd and Nordström [17]. The far-field and wall boundary conditions lead to linear stability for SBP–SAT discretizations of the full Navier–Stokes equations.

In conclusion, the finite-volume scheme can be proven to be linearly stable for the Navier–Stokes equations on structured grids and for the Euler equations on unstructured grids. Furthermore, numerical studies indicate that this holds for the Navier–Stokes equations on unstructured meshes as well.

3. Time integration and splitting algorithm

The semi-discrete finite-volume scheme described above leads to a large system of ODEs. To discretize this system in time, we use the IMEX Runge–Kutta (IMEX-RK) time integrator of Le and Moin [10]. For a general ODE system $\vec{\phi}_t = f(\vec{\phi}, t)$, with $f = f^e + f^i$ denoting a formal splitting into an explicit and implicit part, the three-step time-advancement scheme can be written as

$$\frac{\vec{\phi}^k - \vec{\phi}^{k-1}}{\Delta t} = \gamma_k f^e(\vec{\phi}^{k-1}) + \zeta_k f^e(\vec{\phi}^{k-2}) + \alpha_k f^i(\vec{\phi}^{k-1}) + \beta_k f^i(\vec{\phi}^k), \tag{6}$$

where $k = 1, 2, 3$ denotes the substep number. ($k-2$ is ignored for $k = 1$.) $\vec{\phi}^0$ and $\vec{\phi}^3$ are the solution vectors at time steps n and $n+1$, and Δt is the time-step size. The coefficients γ_k, ζ_k, α_k, and β_k are given in Table 1.

Stability analysis of IMEX-RK schemes has been carried out for the scalar case, $\phi_t = (\lambda^i + \lambda^e)\phi$. For a general system of equations, e.g. $\vec{\phi}_t = K\vec{\phi}$, stability would follow from the scalar results if the system matrix is diagonalizable and spectrally decomposed into explicit and implicit parts, i.e., $K = S\Lambda S^{-1} = S(\Lambda^e + \Lambda^i)S^{-1} = S\Lambda^e S^{-1} + S\Lambda^i S^{-1} = K^e + K^i$, where Λ, Λ^e, and Λ^i are diagonal matrices, S is a matrix whose columns are eigenvectors of K, and K^e and K^i are the explicit and implicit parts, respectively. However, this requires knowledge of the eigenvectors and eigenvalues, which are not known (or computationally too expensive to obtain) in the general case. Below, we propose an algorithm that splits the problem into explicit and implicit parts by ensuring that the spectral radius of the explicit matrix does not violate the CFL condition. In this matter, our analysis is similar to previous work for time advancement of advection–diffusion equations, in which the diffusion terms are advanced implicitly in time while the convection terms are marched in time using an explicit time integrator [29]. We acknowledge the fact that this is not a rigorous stability proof because the resulting explicit and implicit parts are not necessarily diagonalizable and more importantly they are not necessarily simultaneously diagonalizable. Proving stability for an IMEX-RK scheme when applied to a general system of equations is tremendously difficult and to the best of the authors' knowledge, there is no such a proof. However, previous studies (e.g. see [10,11,15]) as well as the current one show that the eigenvalues estimation works well in practice.

3.1. The splitting algorithm

We introduce the splitting algorithm using the linear variable coefficient advection–diffusion equation:

$$\frac{\partial \phi}{\partial t} + \alpha(x)\frac{\partial \phi}{\partial x} = \frac{\partial}{\partial x}\left(\beta(x)\frac{\partial \phi}{\partial x}\right) \quad x \in [0, L], \tag{7}$$

with periodic boundary conditions and $\beta(x) \geqslant 0$. A discretization $\{x_j\}$ of the x-axis and $\Delta_j^+ = x_{j+1} - x_j$, $\Delta_j^- = x_j - x_{j-1}$ and $\Delta_j = (\Delta_j^+ + \Delta_j^-)/2$ is introduced. We approximate the convection term as

$$\alpha(x)\frac{\partial(\phi)}{\partial x}\bigg|_j \approx \alpha(x_j)\frac{\phi_{j+1} - \phi_{j-1}}{2\Delta_j} \quad j = 1, \ldots, N, \tag{8}$$

and the second derivatives as

$$\frac{\partial}{\partial x}\left(\beta(x)\frac{\partial \phi}{\partial x}\right)\bigg|_j \approx \frac{\beta_j + \beta_{j+1}\Delta_j^+\phi_{j+1} - \left[\frac{\beta_j+\beta_{j+1}}{\Delta_j^+} + \frac{\beta_j+\beta_{j-1}}{\Delta_j^-}\right]\phi_j + \frac{\beta_j+\beta_{j-1}}{\Delta_j^-}\phi_{j-1}}{\Delta_j}. \tag{9}$$

Let $\vec{\phi} = (\phi_1, \ldots, \phi_N)^T$ and introduce the diagonal matrix P with positive components $P_{ii} = \Delta_i$. Then the scheme can be written as

$$\vec{\phi}_t = K\vec{\phi} = P^{-1}M\vec{\phi}. \tag{10}$$

This scheme is energy-stable [19,20], implying that $M + M^T$ is negative semi-definite and $\|\vec{\phi}\|_P = \vec{\phi}^T P\vec{\phi}$ is bounded by the initial data.

Table 1
Coefficients of the RK time-advancement scheme of Le and Moin [10].

k	γ_k	ζ_k	α_k	β_k
1	$\frac{8}{15}$	0	$\frac{4}{15}$	$\frac{4}{15}$
2	$\frac{5}{12}$	$-\frac{17}{60}$	$\frac{1}{15}$	$\frac{1}{15}$
3	$\frac{3}{4}$	$-\frac{5}{12}$	$\frac{1}{6}$	$\frac{1}{6}$

Before defining the splitting algorithm, we show that for the system $\vec{\phi}_t = K\vec{\phi} = P^{-1}M\vec{\phi}$, the eigenvalues of the discretization matrix K have non-positive real parts. Because we are considering energy-stable schemes and $M + M^T$ is negative semi-definite, we have $(\vec{\phi}^T P\vec{\phi})_t = \vec{\phi}^T(M + M^T)\vec{\phi} \leqslant 0$ for any solution vector $\vec{\phi}$. By assuming that K has an eigenvalue λ^+ such that Real $(\lambda^+) > 0$ and has a corresponding eigenvector $\vec{v}^+$, then $\vec{v}_t^+ = \lambda^+\vec{v}^+$ and $(\vec{v}^{+,T}P\vec{v}^+)_t = \lambda^+\vec{v}^{+,T}P\vec{v}^+ > 0$, which contradicts the energy stability. Hence, the eigenvalues of K do not have positive real parts.

The objective is to split the system of ODEs (10) into $K = K^e + K^i$, such that K^e has a small spectral radius. Then K^i is the remaining stiff part with a potentially large spectral radius. Because eigenvalue computation is extremely costly for large systems, we propose to base our splitting algorithm on the Gerschgorin theorem. Based on this theorem, all the eigenvalues of the $N \times N$ matrix $K = [k_{ij}]$ are located in the union of discs Ψ_i (in the complex plane $\mathbf{C}$), i.e., $\lambda \in \cup_{i=1}^N \Psi_i$, where λ is any eigenvalue of the matrix K, $\Psi_i = \{z \in \mathbf{C} : |z - k_{ii}| \leqslant \Re_i(K)\}$, and $\Re_i(K) = \sum_{j=1, j\neq i}^N |k_{ij}|$. The schematic of the discs Ψ_i and the corresponding radii $\Re_i(K)$ are shown in Fig. 2.

We define a set of radii $R_i(K) = |k_{ii}| + \Re_i(K)$ and the *Gerschgorin radius* $R_G(K) = \max_i R_i(K)$. It should be noted that $R_G(K)$ dominates the spectral radius of matrix K (denoted by $R_S(K)$), i.e., $R_G(K) > R_S(K)$. In other words, the disc $\{z \in \mathbf{C}: |z| \leqslant R_G(K)\}$ will include all the eigenvalues of matrix K. If $K^i = 0$, the IMEX-RK scheme (6) would be a purely explicit scheme and admit a maximal spectral radius R^e. In that case, by choosing Δt such that $R_G(K) < R^e/\Delta t$, we obtain a stable discretization (see [30]) as is demonstrated in Fig. 2. Based on this observation, for any given Δt, we propose the Row-Splitting algorithm by estimating the eigenvalues of K as

```
for i = 1, N
  calculate R_i(K)
  if (R_i ⩾ R^e/Δt) then
    move the entire row i of K to K^i
  else
    move the entire row i of K to K^e
end for
```

which will guarantee that $R_G(K^e) < R^e/\Delta t$. The algorithm can be written in matrix form as

$$K^e = EK \quad \text{and} \quad K^i = (I - E)K,$$

where E is an $N \times N$ diagonal matrix with

$$E_{ii} = \begin{cases} 1 & R_i < R^e/\Delta t \quad (\text{row } i \text{ is treated explicitly}), \\ 0 & R_i \geqslant R^e/\Delta t \quad (\text{row } i \text{ is treated implicitly}). \end{cases} \tag{11}$$

It is also possible to prove that the real parts of the eigenvalues of both K^i and K^e are non-positive. Because both K^i and K^e are defined similarly, it suffices to study K^e. Define $M^e = PK^e$ and note that $M^e = EM$ because P commutes with E. Consider a non-zero eigenvalue λ^e and a corresponding eigenvector v of the matrix M^e. By definition E is a projection, i.e., $E^2 = E$. Hence, $\lambda^e Ev = EM^e v = E^2 Mv = EMv = M^e v = \lambda^e v$ and $Ev = v$. Moreover,

$$v^H \lambda^e v = v^H M^e v = v^H EM v = v^H EME v, \tag{12}$$

and

$$(\lambda^e + \bar{\lambda}^e) v^H v = (Ev)^H (M + M^T) Ev \leqslant 0. \tag{13}$$

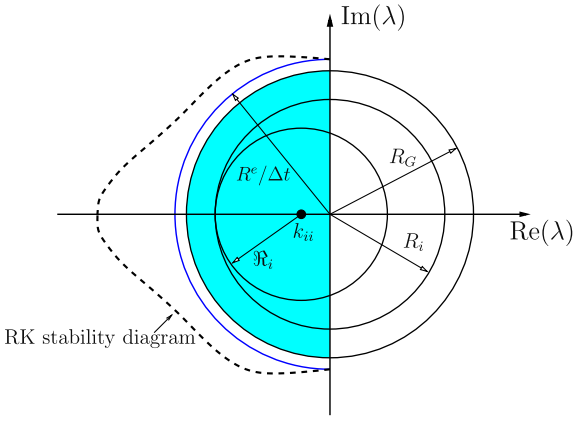

Fig. 2. Schematic of the Gerschgorin's theorem, eigenvalue approximation, and RK stability diagram. The stability diagram (----) is scaled by $1/\Delta t$. The blue semi-circle with radius $R^e/\Delta t$ is the largest semi-circle with center at the origin of the complex plane which is circumscribed by the stability region of the RK time integrator. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Because v is an eigenvector with l_2-norm 1, non-positiveness of the real part of λ^e follows from negative semi-definiteness of $M + M^T$.

In conclusion, for a given Δt, the aforementioned splitting algorithm will guarantee that all the eigenvalues of the matrix K^e lie inside the semi-disk $\{z \in \mathbf{C}: |z| \leqslant R^e/\Delta t \,\&\, \mathrm{Re}\,(z) \leqslant 0\}$ depicted in cyan in Fig. 2.

3.1.1. Numerical example

To test the algorithm, we use the advection–diffusion Eq. (7) and we set $\alpha = 1$ and vary β to control the dissipation. First, we consider the following three cases:

(a) $\beta = 3.6 \times 10^{-5}$,
(b) $\beta = 1.7 \times 10^{-3}$,
(c) $\beta = 3.6 \times 10^{-3}$.

For a small β (Case *a*), the real parts of the eigenvalues of the matrix K are small. Maximum value of the real parts of the eigenvalues of the Case *b* is of the same order as the maximum value of their imaginary parts, whereas for Case *c*, maximum real parts is an order of magnitude larger than the maximum imaginary parts (see Fig. 3).

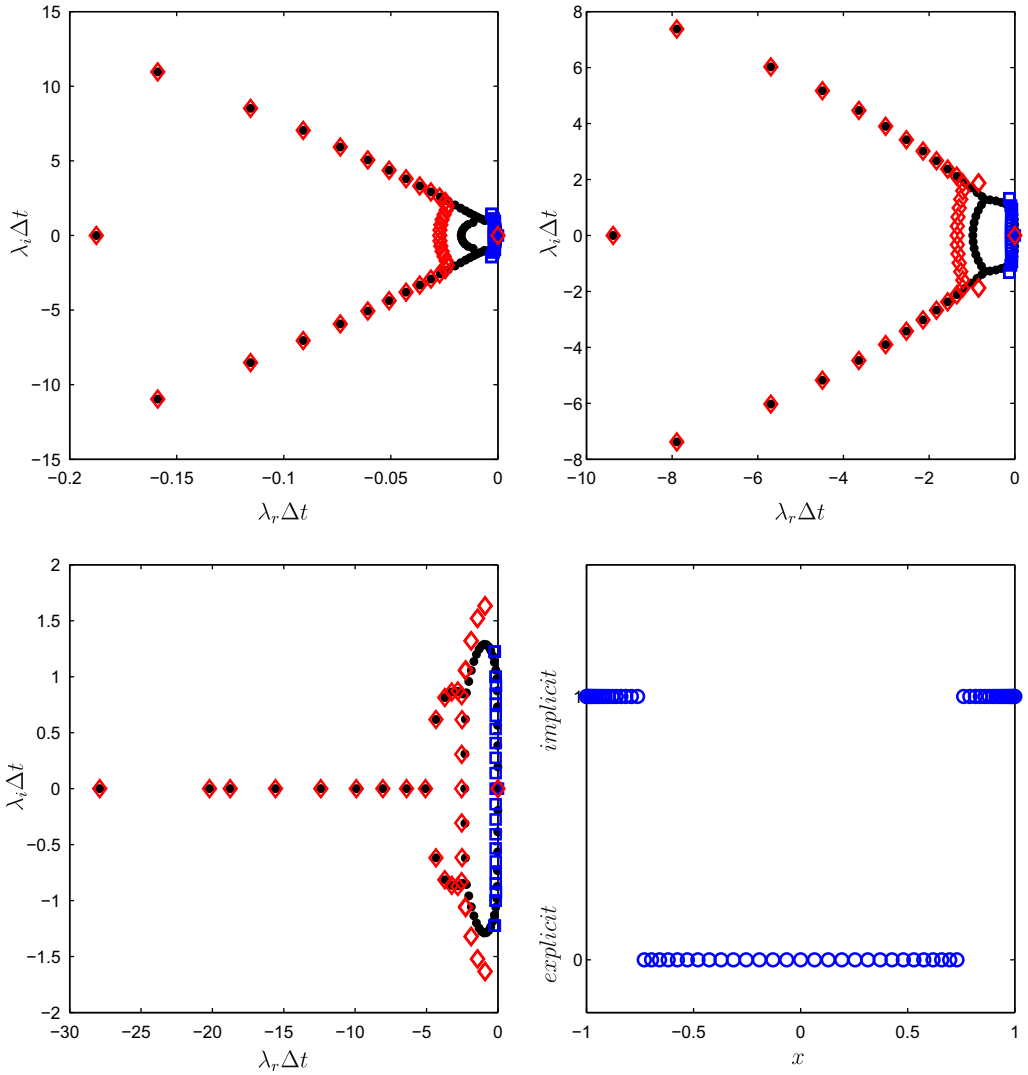

Fig. 3. Eigenvalue distribution of the advection–diffusion equation using a hyperbolic tangent grid stretching. Top-left: Case *a*; top-right: Case *b*; bottom-left: Case *c*; and bottom-right: implicit–explicit grid points distribution for Case *a*. 55% of the nodes are advanced implicitly. For the top figures and the bottom-left, •: eigenvalues of the original system, □: eigenvalues of the explicit fluxes, and ◇: eigenvalues of the implicit fluxes.

For all the three cases R^e is set to $\sqrt{3}$ (see [10]) and $\Delta t = 6.2 \times 10^{-2}$. A total of 64 grid points are used to discretize the domain $[-1,1]$. The stretching parameter is based on a hyperbolic tangent function to cluster points at the boundaries. The ratio of the largest grid spacing to the smallest one is set to 13.3. Eigenvalues of both the original and split system are computed and shown in Fig. 3 (note that the scales of the axes are different). The eigenvalues of the explicit matrix are clustered near the origin, whereas the implicit matrix has eigenvalues with large magnitudes (of the same order as the original system).

To study the effect of irregular grids, we recompute Case *b*. To this end, we consider an extreme case for which a random grid with a ratio of the maximum to minimum grid spacing of 20 is used. The eigenvalue distribution of both the original and the split systems is shown in Fig. 4. Again, the method is capable of clustering the eigenvalues of the explicit matrix near the origin, which indicates that the algorithm is working. Note that in all the cases, the eigenvalues are non-positive as proved earlier in this section.

Lastly, to study the effect of diffusion (numerical/physical) on the stability of the RS-IMEX method, we consider a marginally stable case. Here, we use a uniform grid with 64 grid points and we set $\beta = 0$. For this problem, the discretization (8) does not contain any numerical diffusion and thus, all the eigenvalues of the discretized system are imaginary. If the proposed time splitting scheme is applied to this problem, depending on the time-step size all the grid points will be either explicit or implicit. However, we can artificially assign the implicit/explicit attribution and test the method with a manufactured implicit–explicit splitting. For this purpose, the time step size is set to $\Delta t = 3.12 \times 10^{-2}$ and implicit time advancement is applied to one-third of the grid points. Fig. 5 illustrates the implicit grid points distribution and the eigenvalues of the original and the split system. One can see that the eigenvalues of both explicit and implicit matrices have zero real parts. Furthermore, the field is initialized with a sine profile $\phi(t=0) = \sin(-\pi x)$. The equations are advanced in time up to final time $t = 100$. The exact solution is computed as $\phi_{exact} = sin(-\pi(x-t))$. Moreover, we compute the solution at $t = 100$ with a significantly small time-step size $\Delta t = 3.12 \times 10^{-4}$ and we denote it by $\bar{\phi}$. Fig. 5 shows the two errors $\phi - \phi_{exact}$ and $\phi - \bar{\phi}$ at $t = 100$ (50 flow throughout the domain). Note that $\phi - \bar{\phi}$ is dominated by the time discretization errors. A slight non-smoothness in this error is due to the mixed implicit–explicit time-advancement method, however, it is of $O(\Delta t^2)$. The results show that the solution stays bounded for a long integration time and, therefore, the method is stable in absence of any physical and grid-induced numerical diffusion (note that numerical diffusion caused by the time-advancement scheme is still present).

3.2. Implicit–explicit splitting for the Navier–Stokes equations

Consider the approximation (3) of the Navier–Stokes equations, compactly written as

$$\frac{\partial \vec{U}}{\partial t} = \vec{H}(\vec{U}), \tag{14}$$

where $\vec{U} = (\vec{U}_1^T, \ldots, \vec{U}_N^T)^T$ is the solution vector and N the number of grid points. Similarly, the right-hand side $\vec{H} = (\vec{H}_1^T, \ldots, \vec{H}_N^T)^T$ is the approximate gradient of the flux vector

$$\vec{H}_k = -\frac{1}{V_{\Omega_k}} \sum_f \sum_{j=1}^{3} \left(\vec{F}_j^f - \frac{1}{Re} \vec{G}_j^f \right) \vec{n}_j^f. \tag{15}$$

As before, we introduce $\vec{H} = \vec{H}^e + \vec{H}^i$. The stiffness is estimated from the eigenvalues of the flux Jacobian matrix. The Jacobian matrix J consists of 5×5 blocks $\mathcal{J}^{ij}$ such that

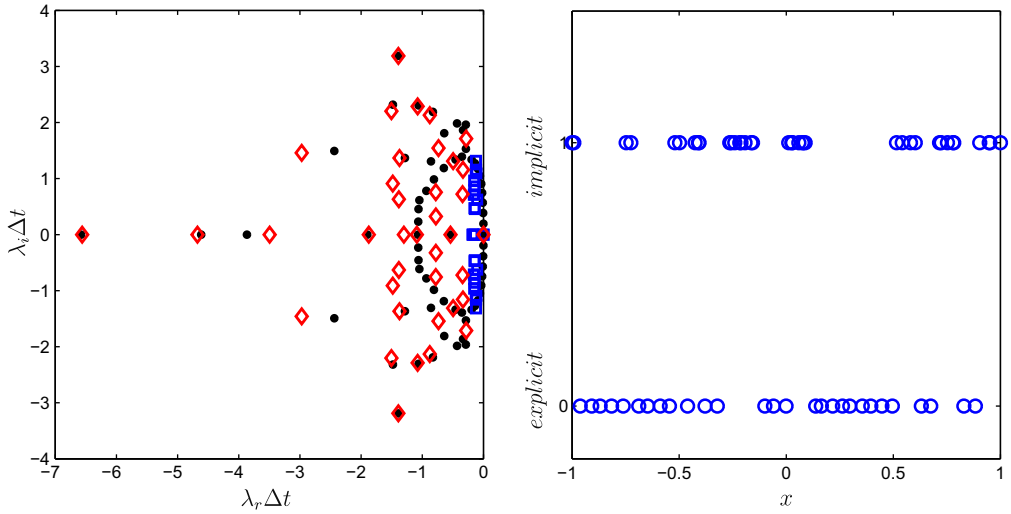

Fig. 4. Advection–diffusion equation with a random grid. Left: eigenvalues of both original and split system. Symbols denote the same quantities as in Fig. 3. Right: grid, 57% of the nodes are implicit.

$$J = \frac{\partial \vec{H}}{\partial \vec{U}} = [\mathcal{J}^{ij}], \quad \mathcal{J}^{ij} = \frac{\partial \vec{H}_i}{\partial \vec{U}_j}, \quad i = 1, \ldots, N; \quad j = 1, \ldots, N. \tag{16}$$

We find that the final matrix J is sparse because most $\mathcal{J}^{ij}$ are 0.

We generalize the splitting algorithm proposed in Section 3.1 to the Navier–Stokes equations by applying the proposed splitting method to the Jacobian matrix J. However, we simplify the algorithm and do not consider every row in the matrix individually. Instead, we move the entire flux $\vec{H}_i$, corresponding to the grid point i, to either the explicit ($\vec{H}^e$) or the implicit ($\vec{H}^i$) part. Note that the flux $\vec{H}_i$ corresponds to the five rows $5i-4,\ldots,5i$ in the Jacobian matrix J (these five rows are called *block row i*). To this end the block radii are calculated as

$$\widehat{R}_i(J) = \max_{l \in \{1,\ldots,5\}} \left\{ \sum_{j=1}^{N} \sum_{m=1}^{5} |\mathcal{J}^{ij}_{lm}| \right\} \quad i = 1, \ldots, N, \tag{17}$$

where $\mathcal{J}^{ij}_{lm}$ denotes the components of $\mathcal{J}^{ij}$. For a given Δt, we propose the following splitting algorithm:

```
for all grid points i from 1 to N
  calculate the block radius R̂_i
  if (R̂_i ⩾ R^e/Δt) then
    move the flux H⃗_i to implicit portion H⃗^i
  else
    move the flux H⃗_i to explicit portion H⃗^e
end for
```

that ensures that the Jacobian matrix of $\vec{H}^e$ has a spectral radius smaller than $R^e/\Delta t$. In matrix form, the algorithm reduces to $\vec{H}^e = (E \otimes I_5)\vec{H}$ where E is an $N \times N$ diagonal matrix defined in (11) and I_5 is the 5×5 identity matrix. Finally, using the proposed algorithm, the semi-discrete equations are split into

$$\frac{\partial \vec{U}}{\partial t} = \vec{H}^e(\vec{U}) + \vec{H}^i(\vec{U}) \tag{18}$$

and are advanced using the RK time integrator in (6).

3.3. Practical considerations

LES and DNS of flows of practical interest involve millions of grid points requiring efficient parallel algorithms. In this section, we briefly discuss the key elements in the application of the method to large-scale computations.

3.3.1. Further element reduction from implicit fluxes

Thus far we have treated an entire *block row i* in the Jacobian matrix, which corresponds to the flux $\vec{H}_i$, as either explicit or implicit. We call any grid point *j a neighbor* to *i* if $\mathcal{J}^{ij} \neq 0$. In its present form, the proposed algorithm does not take directional stiffness (such as wall-normal derivatives in a boundary layer) into account because the implicit/explicit attributions are associated with the fluxes and the contributions from all neighbors of a grid point are treated in the same way (either implicit or explicit).

Hence, to refine the algorithm we remove terms from implicit fluxes that contribute negligibly to the stiffness. We define

$$\widetilde{R}_{ij} = \widetilde{R}(\mathcal{J}^{ij}) = \max_{l \in \{1,\ldots,5\}} \left\{ \sum_{m=1}^{5} |\mathcal{J}^{ij}_{lm}| \right\}, \tag{19}$$

which represents the stiffness contribution to grid point i from grid point j. To reduce the number of elements in the implicit part, we modify the algorithm as:

```
for all grid points i from 1 to N
  calculate the block radius R̂_i
  if (R̂_i ⩾ R^e/Δt) then
    for all grid points j that are neighbors of i
    if (R̃_ij > R^e/(χΔt)), then
      move the flux term associated with
      grid point j to the implicit portion
    end if
  end for
else
  move the flux H⃗_i to explicit portion H⃗^e
end for
```

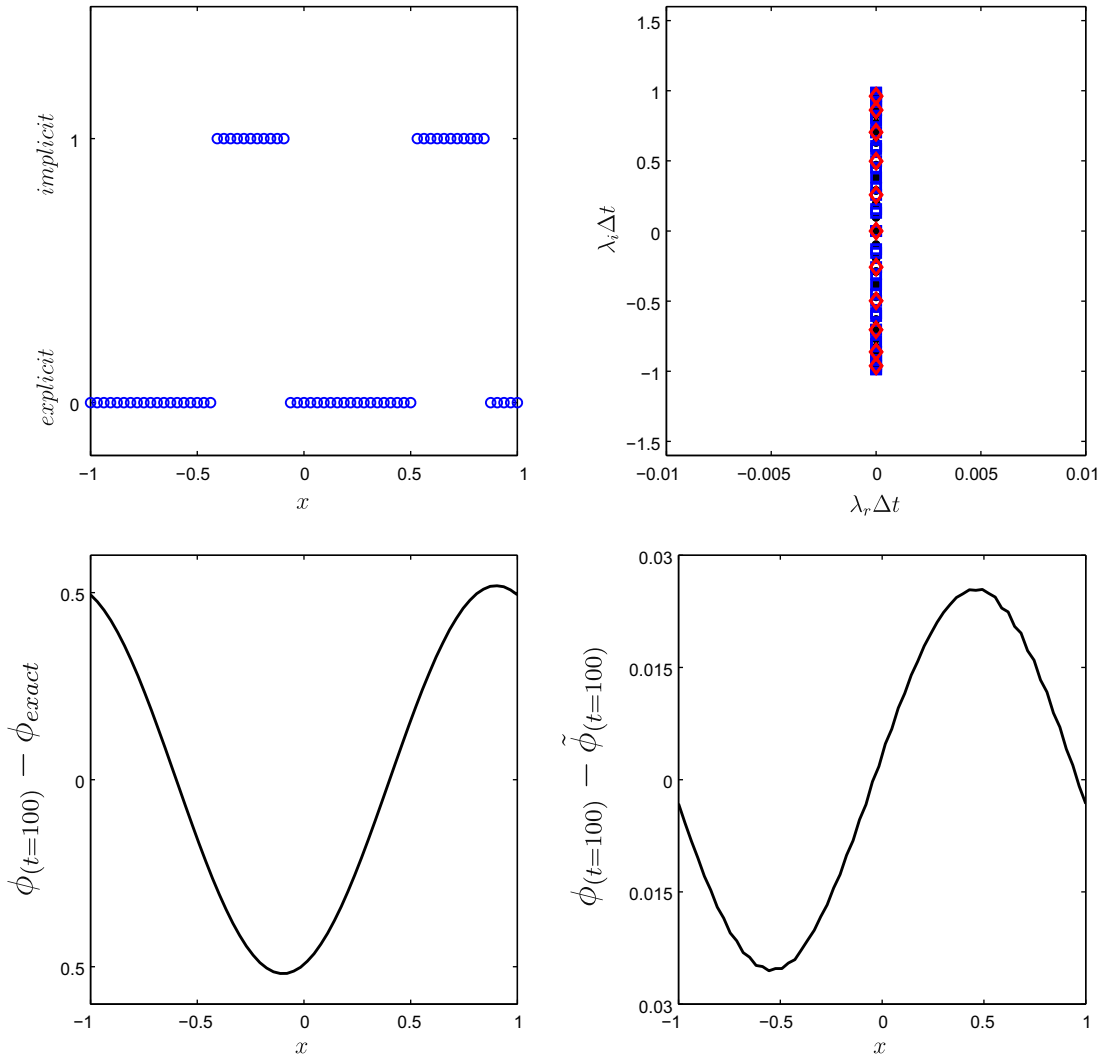

Fig. 5. Manufactured Row-Splitted advection–diffusion equation with $\beta = 0$ on a uniform grid. Top-left: manufactured implicit–explicit grid points distribution; top-right: eigenvalue distribution (symbols denote the same quantities as in Fig. 3); bottom-left: error with respect to exact solution at $t = 100$; bottom-right: error with respect to the computed solution $\tilde{\phi}$ on the same grid but with a significantly smaller time-step size $\Delta t = 3.12 \times 10^{-4}$.

where χ is a grid type- (e.g. hexahedral or tetrahedral) dependent parameter. Another approach would be to choose the p smallest $\widetilde{R}_{ij}$'s such that $\sum_p \widetilde{R}_{ij} \leqslant R^e$. However, this requires more work and more communication between the processors than does our approach with χ. For a hexahedral grid, a suitable value is $\chi = 27$ (the number of neighbors). It is shown in Section 4.5 that the modified algorithm results in a significant element reduction of the implicit fluxes.

3.3.2. Non-linear solver

Due to non-linearity in the flux vector $\vec{H}^i$, a non-linear system of equations must be solved at each Runge–Kutta stage. Here, we use an incomplete Newton method that does not update the Jacobian matrix during the Newton iterations. The bi-conjugate gradient solver of [31] with a block-Jacobi preconditioner (e.g. see [32]) is used to solve the linear system. A more sophisticated linear solver including proper preconditioners for different Mach numbers is a subject of future work and has not been studied here.

3.3.3. Domain decomposition

For the proposed RS-IMEX algorithm with two different time-advancement schemes, the memory and work required for every implicit grid point is more than that required for an explicit one. On parallel computers, the computational domain must be decomposed such that both computational work and memory are balanced between the processors. However, it is difficult to estimate the amount of work required for an implicit grid point because the convergence properties, and as a result the required work depends on the flow, the grid, and even the time-step size.

We therefore apply a dual-constraint partitioning algorithm. Here, we use ParMetis libraries [34], which are capable of multi-constraint partitioning to decompose the domain among the processors. Through the constraints, the method tries to balance both the number of grid points per processor and the number of implicit grid points per processor. At the same time, it minimizes the edge-cuts at the boundaries between the processors. This approach does not require any information about the work ratio between an explicit and implicit grid point and balances both work and memory simultaneously and therefore, it is flow and time-step size independent. The main drawback of this approach is the fact that it might generate disjointed partitions that will increase the edge-cuts at the boundaries between processors compared with those of a single-constraint partitioning algorithm. This increase in the edge-cuts will result in a longer communication time. However, Karypis and Kumar [33] showed that the increase is not more than 20–30% of the single-constraint partitioning edge-cut for a wide range of number of partitions. Therefore, the communication overhead due to this increase in the edge-cut will not dominate the speedup as long as the total communication time due to message passing at the interface between the processors is smaller than the computation time. Our numerical tests in Sections 4.5 and 4.6 confirm this fact and show that the method reasonably scales up to large number of processors.

4. Verification and performance of the method

The 3D unstructured finite-volume scheme has been implemented in a Message Passing Interface (MPI) code capable of running on an arbitrary number of processors with shared or distributed memory. Below we verify and show performance of the method on a number of different test cases.

We begin by explaining some terminology for an unstructured grid (such as in Fig. 1).

CFL: is defined as

$$\mathrm{CFL} = \frac{\Delta t}{2} \max_{\Omega_k} \left\{ \frac{1}{V_{\Omega_k}} \sum_f (|\vec{u}^f . \hat{\vec{n}}^f| + c^f) A^f \right\},$$

where c is the local sound speed, $\vec{u}$ is the velocity vector, and A^f is the area of the face $\partial\Omega_k^f$.

Implicit node: A grid point for which some part of its flux is in the implicit portion.

Implicit fraction: The fraction of implicit nodes to the total number of grid points.

Nodal implicit fraction: For a point i, the fraction of neighboring grid points that have non-zero contribution to its implicit flux.

4.1. Two-dimensional propagating vortex

To verify the accuracy of the RS-IMEX finite-volume scheme, we use an exact vortex solution to the Euler equations as

$$u = M_\infty \left(\cos\theta - \frac{\Gamma}{2\pi}(y - y_0 - \tilde{v}t) e^{\frac{f(x,y,t)}{2}} \right),$$
$$v = M_\infty \left(\sin\theta + \frac{\Gamma}{2\pi}(x - x_0 - \tilde{u}t) e^{\frac{f(x,y,t)}{2}} \right),$$
$$\rho = \left(1 - \frac{\Gamma^2(\gamma-1)M_\infty^2}{8\pi^2} e^{f(x,y,t)} \right)^{\frac{1}{\gamma-1}},$$
$$p = \frac{1}{\gamma} \left(1 - \frac{\Gamma^2(\gamma-1)M_\infty^2}{8\pi^2} e^{f(x,y,t)} \right)^{\frac{\gamma}{\gamma-1}},$$

where γ is the gas specific heat ratio; c_∞ and ρ_∞ are the reference velocity and density, respectively; Γ is the non-dimensional circulation; M_∞ is the free stream Mach number; $\tilde{u} = M_\infty \cos\theta$ and $\tilde{v} = M_\infty \sin\theta$ are the propagation velocities in x and y directions, respectively; $f = 1 - (x - x_0 - \tilde{u}t)^2 - (y - y_0 - \tilde{v}t)^2$; and (x_0, y_0) is the location of the vortex at $t = 0$.

A vortex with radius one, a freestream Mach number, $M_\infty = 0.5$ and non-dimensional circulation, $\Gamma = 5.65$ is placed at $(x_0 = -1, y_0 = -1)$ and is propagated diagonally to the point $(x = 1, y = 1)$, a distance equal to 2.83 times the vortex core radius (see Fig. 6). The initial and boundary data are provided by the exact solution. The time-step size is chosen such that CFL = 3, which results in a non-zero implicit fraction. The grid is finer close to $(x = 0, y = 0)$ (see Fig. 7), and thus the implicit nodes are clustered close to the origin. As the vortex moves, the characteristic velocities u and c vary in time and as a result the stiffness associated with each grid point changes. This change confirms the fact that not only the geometric stiffness is taken into account, the flow stiffness owing to change in the CFL number is also captured.

In order to perform a convergence study, the grid is refined homogeneously in all directions. The CFL number is kept constant such that the implicit fraction is kept roughly constant among different grid levels (see Fig. 8). This ensures that both the spatial and temporal convergence rates are measured. Indeed, we observe a second-order convergence rate in L_2 and L_∞ (see Fig. 7).

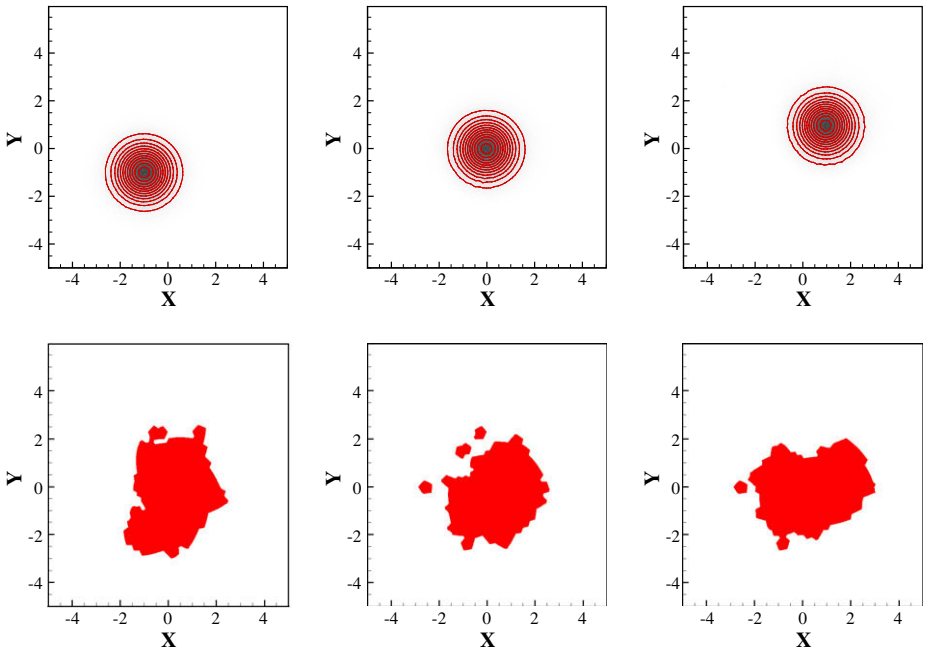

Fig. 6. Mixed implicit–explicit time-advancement. Top: density contours; bottom: implicit regions at different times. The scheme is able to stably capture and mark the implicit regions.

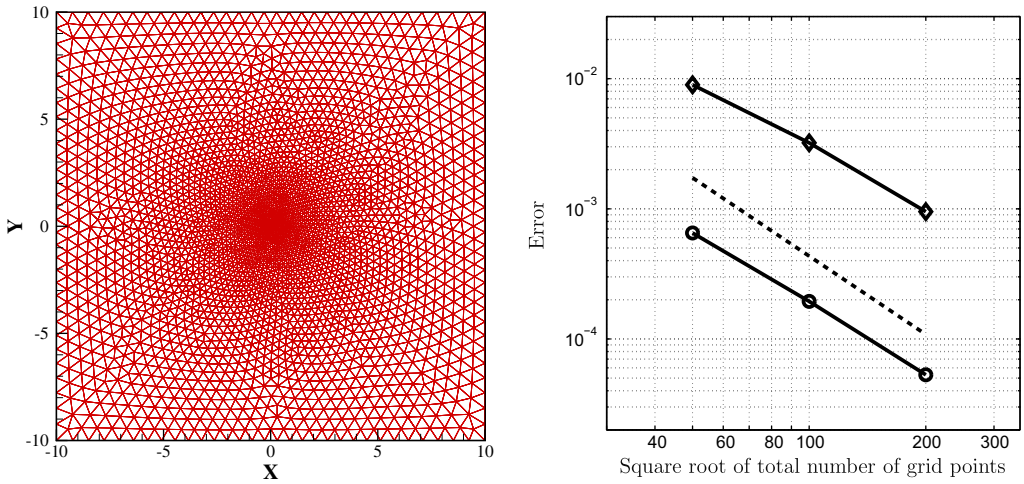

Fig. 7. Convergence study for the 2D vortex. Left: grid; right: error in pressure. For the right figure, ∘: L_2 error; ♢: L_∞ error; —-: second-order convergence.

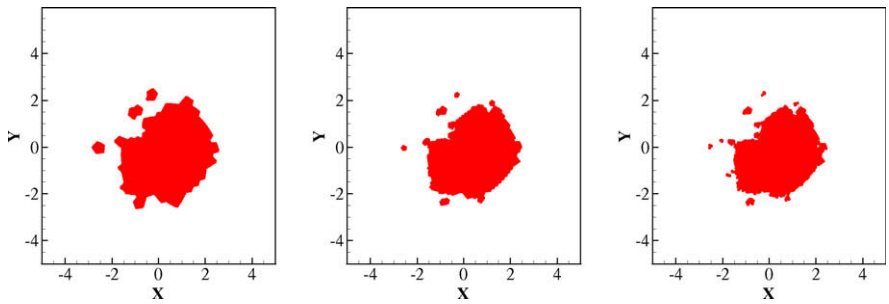

Fig. 8. Implicit mapping for the three grids used for convergence study. Left: coarse grid; middle: medium grid; right: fine grid.

To investigate the stability of the RS-IMEX method when applied to the Euler equations in absence of grid-induced numerical diffusion, we recompute the vortex on a uniform Cartesian grid with 100×100 elements. The vortex is initially located at ($x = -2$, $y = 0$), propagates in x direction ($\theta = 0$) with velocity $M_\infty = 0.5$ and finally arrives at ($x = 2$, $y = 0$). Periodic boundary conditions are imposed in all directions to eliminate the effect of dissipation from boundary conditions. In order to have more distinct implicit–explicit regions, a strong vortex with $\Gamma = 11.31$ is used. The time step size chosen such that CFL = 1.3. Since there is not a wide range of eigenvalues for this problem, larger time-step sizes will result in a fully implicit domain. Fig. 9 shows both the density contours and implicit regions at both the beginning and the end of the simulation. The results verify that the scheme can stably propagate the vortex and hence, does not rely on any grid-induced numerical diffusion (note that numerical diffusion caused by skew-symmetric form and the time-advancement scheme are still present).

4.2. Boundary layer transition and growth of Tollmien–Schlichting waves

Sharp near-wall gradients due to the no-slip condition require small grid spacing in the wall-normal direction, necessitating implicit treatment. At the same time, the wall-parallel grid resolution can be orders of magnitude larger and, hence, the fluxes associated with those directions can be marched explicitly in time.

We simulate the growth of Tollmien–Schlichting (TS) waves in a flat-plate laminar boundary layer. The computational domain is chosen such that $Re_x = U_\infty x/\nu$ varies from 15,000 to 414,000. The initial mean field is calculated using the similarity solution (see e.g. [36]) with freestream Mach number 0.2 and adiabatic wall boundary conditions. To damp the disturbances at the inflow/outflow boundaries, a sponge layer is applied and the flow is penalized against the similarity solution. The TS waves are generated using the Fasel and Konzelmann method [37] in which they introduced a permeable wall over the range $Re_x = 0.9 \times 10^5$ ($x = x_1 = 0.9$) to $Re_x = 1.1 \times 10^5$ ($x = x_2 = 1.1$) with the prescribed velocity $v/c_\infty = A\hat{v}(x)\sin(2\pi ft)$ where

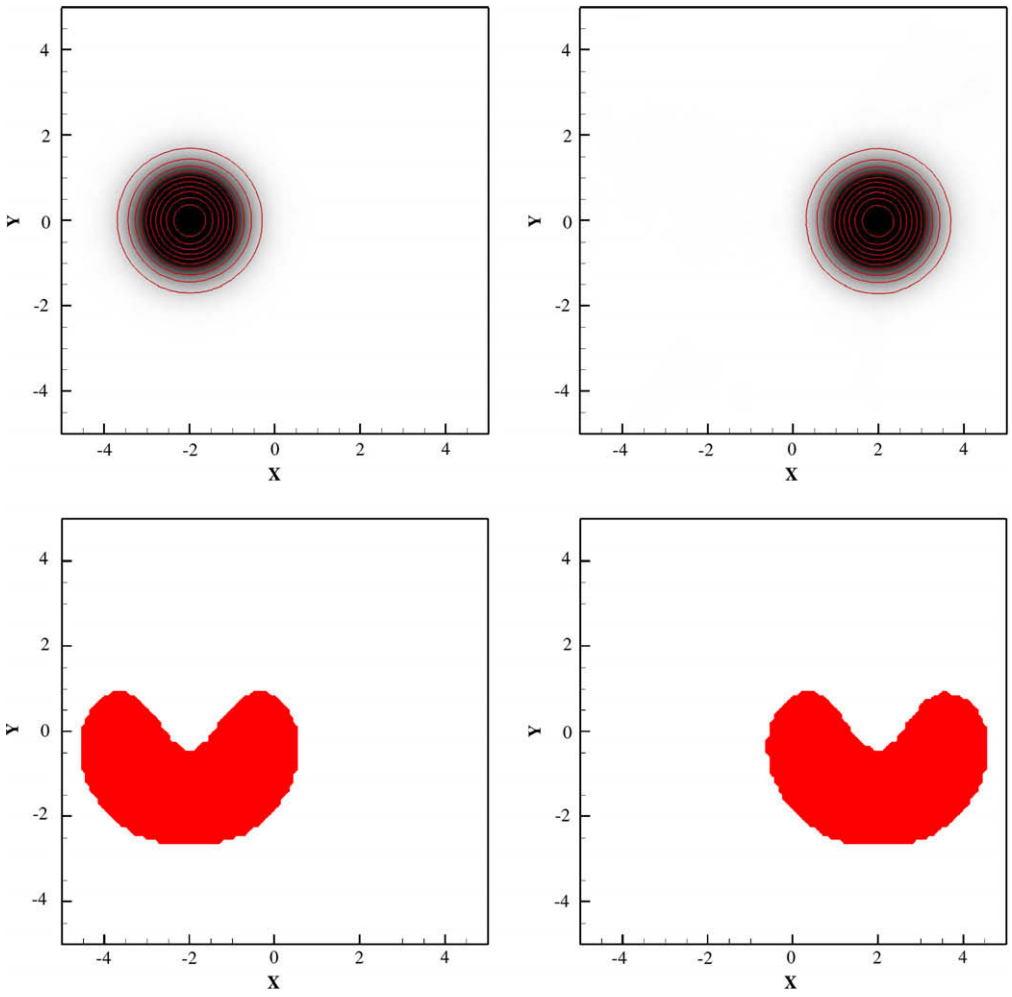

Fig. 9. Mixed implicit–explicit time-advancement on a uniform grid. Top: density contours; bottom: implicit regions; left: initial vortex; right: vortex after propagating four length units in x direction.

Table 2
Implicit fraction as a function of CFL number for the boundary layer transition simulation.

CFL	1	2	5	10
Implicit fraction (%)	0.99	1.98	28.72	49.54

$$\hat{v}(x) = 20.25\xi^3 - 35.4375\xi^4 + 15.1875\xi^5,$$

with

$$\xi = \begin{cases} \frac{x-x_1}{x_m-x_1} & \text{for } x_1 < x < x_m \\ \frac{x_2-x}{x_2-x_m} & \text{for } x_m < x < x_2 \end{cases}, \quad x_m = \frac{x_1+x_2}{2},$$

$A = 2.0 \times 10^{-4}$, and $2\pi f = 1.4 \times 10^4 U_\infty^2/\nu_\infty$. In the streamwise direction, 400 grid points are uniformly distributed, whereas 101 grid points are used in wall-normal direction, y. At the inflow where $Re_x = 15{,}000$, 16 grid points are inside the boundary layer, and the first grid point in the y-direction is at $\Delta y = 5\%\delta_{99}$ where δ_{99} is the boundary-layer thickness. The disturbances are introduced at $t = 0$ and the simulation is carried out over 30 periods of the prescribed velocity at the permeable wall. Throughout the final period, 100 samples equally spaced in time have been collected and the growth rate α of the disturbances is calculated, based on the inner peak of the streamwise velocity as defined in [37].

Table 2 shows the implicit fractions as a function of CFL number. As the CFL increases, the number of grid points for which some part of their fluxes is advanced implicitly in time increases, resulting in higher implicit fractions. These grid points, denoted as implicit nodes, are clustered near the wall. Note that at CFL = 1, only grid points on the boundary-layer wall are implicit nodes. This implicit treatment is required due to the wall penalty terms used to impose no-slip boundary conditions [17]. The added fluxes to impose no-slip conditions at walls introduce local stiffness in the system that requires those fluxes to be advanced implicitly in time.

In addition, the growth rates are plotted in Fig. 10 and are compared with the incompressible analysis [38] and high-order compressible calculations [39]. The current simulation results are in agreement with previous compressible results and are slightly shifted relative to the incompressible analysis. Furthermore, the results with mixed implicit–explicit time-advancement at CFL = 5.2 match those with explicit time-advancement at CFL = 1. For other values of CFL reported in Table 2, the growth rates are almost identical (within 1% of the results at CFL = 1).

4.3. Low Reynolds number flow around a 2D circular cylinder

In this study, we consider the unsteady compressible laminar flow around a 2D circular cylinder with diameter D at $Re_D = U_\infty D/\nu = 100$. At this Reynolds number, the flow behind the cylinder separates and vortex shedding occurs. The free-stream Mach number is sufficiently small ($Ma = 0.1$) to be able to compare the numerical results from the compressible flow simulation with the available incompressible data. The computational domain is a circle of radius $30D$. In [17], the far-field boundary conditions were shown to have a small effect on the computational solution. Quadrilateral mesh cells are used near the cylinder, whereas triangles are used to capture the wake. To resolve the thin laminar boundary layer in front of the cylinder, the minimum radial grid spacing is set to $3 \times 10^{-3}D$. 260 points are used in the circumferential direction. An isothermal wall boundary condition is used on the cylinder wall. Fig. 11 shows the mesh used for the simulation. The simulation started with uniform flow plus random fluctuations to accelerate the onset of the vortex shedding. A snapshot of density contours in the wake of the cylinder is given in Fig. 11. Averages and other quantities are measured once the statistically stationary solution is reached. Table 3 compares the simulation results with the existing experimental data as well as with

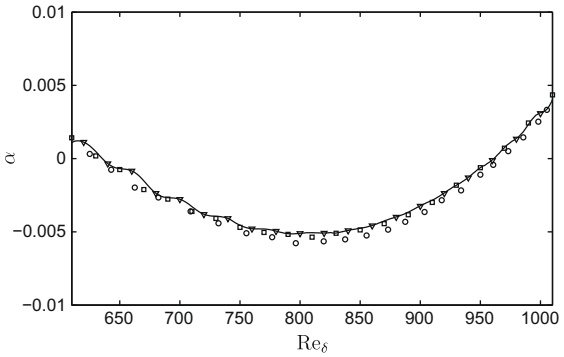

Fig. 10. Growth rates (α) of the TS waves as a function of Reynolds number ($\mathrm{Re}_\delta = U_\infty\delta/\nu$). Solid line: current simulation with CFL = 1; $\bigtriangledown$: current simulation with CFL = 5.2; □: simulation of Nagarajan et al. [39] using a 6th-order compact Pade scheme; ○: incompressible analysis of Gaster [38].

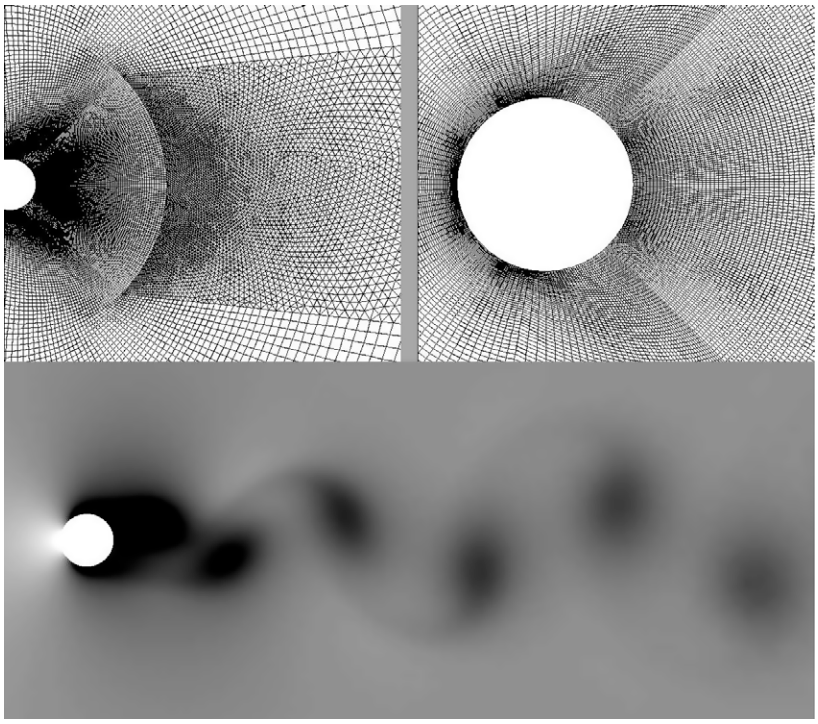

Fig. 11. Top: grid used in the vicinity and in the wake of the cylinder; bottom: vortex shedding behind the cylinder.

Table 3
Simulation results for laminar flow around a 2D circular cylinder and comparison with experimental data as well as incompressible and compressible simulations. C_d: drag coefficient; P_b: back pressure; St: Strouhal number; θ_{sep}: separation angle; L_b: recirculation bubble length.

	C_d	P_b	St	θ_{sep}	L_b
Current simulation	1.347	0.74	0.168	118.6	1.41
Kwon and Choi [40]	1.340		0.166		1.40
Kravchenko and Moin [41]		0.71	0.162	117.3	1.45
Fey et al. [42]			0.165 ± 0.001		
Svärd and Nordström [17]	1.341		0.165		

incompressible and compressible simulations. The results demonstrate that the numerical algorithm is capable of capturing the unsteady flow on mixed grids.

4.4. Homogenous isotropic turbulence

Numerical simulation of homogenous isotropic turbulent flow has proved useful to study the properties of numerical schemes for turbulent flows [3,22,43]. In this section, we validate the method by performing DNS of compressible decaying homogenous isotropic turbulence on both a uniform Cartesian grid and a triply-periodic unstructured mesh with tetrahedral elements.

The domain is taken to be a box of length 2π. The method of Blaisdell et al. [44] is used to generate the initial conditions. At the start of the simulation, the pressure and density are set to be uniform, and incompressible turbulence is added only to the velocity field. The magnitude of the fluctuations are such that they follow the prescribed energy spectrum:

$$E(k) = 16\sqrt{\frac{2}{\pi}}\frac{u_0^2}{k_0}\frac{k^4}{k_0^4}e^{-2\frac{k^2}{k_0^2}}, \tag{20}$$

where k is the wavenumber, and k_0 is the most energetic mode in the initial condition. The initial turbulent Mach number is given by $M_{t_0} = \sqrt{3}u_0/c_0$ where c_0 is the initial sound speed. The Reynolds number based on Taylor microscale (λ) and eddy turnover time (τ) is $Re_\lambda = \frac{2\rho_0 u_0}{\mu_0 k_0}$ where μ_0 is the initial viscosity. Time is non-dimensionalized with $\tau = \lambda/u_0 = 2\sqrt{3}/(k_0 M_{t_0} c_0)$.

First, we validate the method by comparing our results with the DNS results of [22]. For the structured mesh, 64 grid points are used in each direction. The unstructured grid has the same number of points at each edge of the computational

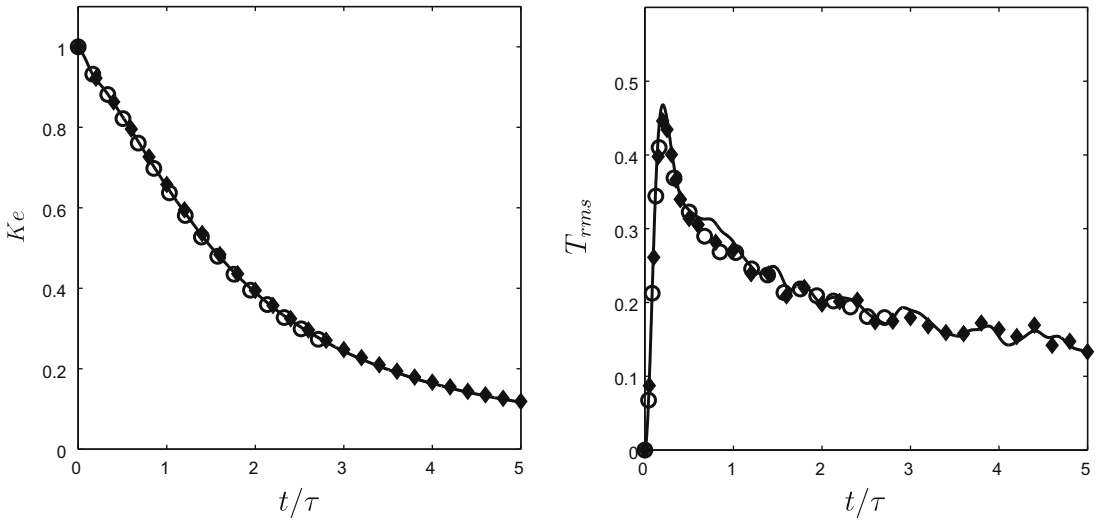

Fig. 12. Time evolution of turbulence quantities. Left: turbulent kinetic energy (Ke) normalized by the initial value; right: temperature fluctuations $T_{rms}/[(\gamma-1)M_{t_0}^2]$; —: structured grid; ♦: unstructured grid; and ○: Honein and Moin [22].

box and the total number of points is set to be almost equal to the structured case. The simulation is performed with $Re_\lambda = 30$, $M_{t_0} = 0.3$, and $k_0 = 4$. The equations are non-dimensionalized with the initial density, temperature, and sound speed. The evolution of the non-dimensional turbulent kinetic energy and the non-dimensional root mean square (rms) of the temperature fluctuations are shown in Fig. 12. The results demonstrate that the present unstructured scheme captures the evolution of the turbulent velocity and temperature fluctuations.

To investigate the effect of the grid on different turbulence eddy sizes, the 3D energy spectra of turbulent kinetic energy at $t/\tau = 5$ are calculated and plotted in Fig. 13. To obtain the spectrum for the unstructured grid, the solution has been linearly interpolated into a uniform Cartesian grid with 64 points in each direction. The energy pile-up at high wavenumbers is likely due to interpolation errors that become more prominent at high wavenumbers.

So far the time-step size has been small enough not to have any implicit regions in the domain. An additional simulation was carried out on the unstructured grid with a CFL number such that the implicit fraction is initially 10%. The time-implicit mapping is updated every 500 time-steps. The 3D energy spectra and the time histories are shown in Fig. 14 and agree well with the previous explicit results. As the turbulence decays, the CFL number decreases thus resulting in a reduction of the implicit fraction.

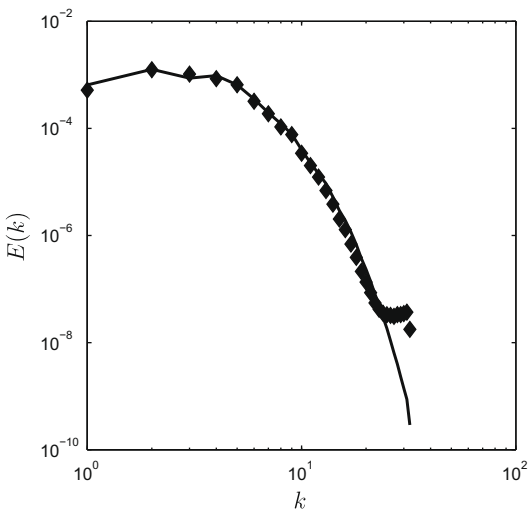

Fig. 13. 3D energy spectra at $t/\tau = 5$. —: structured grid; ♦: unstructured grid.

4.5. Flow around a circular cylinder at $Re_D = 3900$

The objective of this section is to validate our method and study its performance for LES. We perform LES of flow over a circular cylinder at subcritical Reynolds number $Re_D = \frac{U_\infty D}{\nu_\infty} = 3900$, which has been extensively studied for both incompressible [45] and compressible [46] flows. The boundary layer around the cylinder requires a very fine grid, whereas the LES away from the cylinder is less resolution demanding. As a result, implicit time-advancement is needed only near the cylinder walls, and the rest of the computational domain can be advanced explicitly in time.

The simulation is performed with freestream Mach number, $Ma_\infty = 0.4$ on an O-type mesh with radius $24D$ and width πD containing 3.2 million grid points. The time-step, Δt, is chosen to be constant such that the maximum CFL is approximately 16. This time-step results in an implicit fraction of 27%. The nodal implicit fraction is 7/27, meaning that only $7/27 \approx 25\%$ of the neighbors of an implicit node contribute to its implicit fluxes and the fluxes associated with the rest are advanced explicitly in time. The time-implicit mapping is updated 10 times during each shedding cycle. A snapshot of regions with implicit nodes, which are clustered near the cylinder, is given in Fig. 15. The subgrid stress tensor is modeled using the dynamic Smagorinsky [47] with Lilly's modification [48]. Spanwise averaging is used to calculate the model coefficients and they are clipped in case of negative total viscosity. In this simulation, less than 3% of the grid points were clipped, and the maximum ratio of the total viscosity to the molecular viscosity is around 10. A snapshot of the vorticity magnitude and the contribution

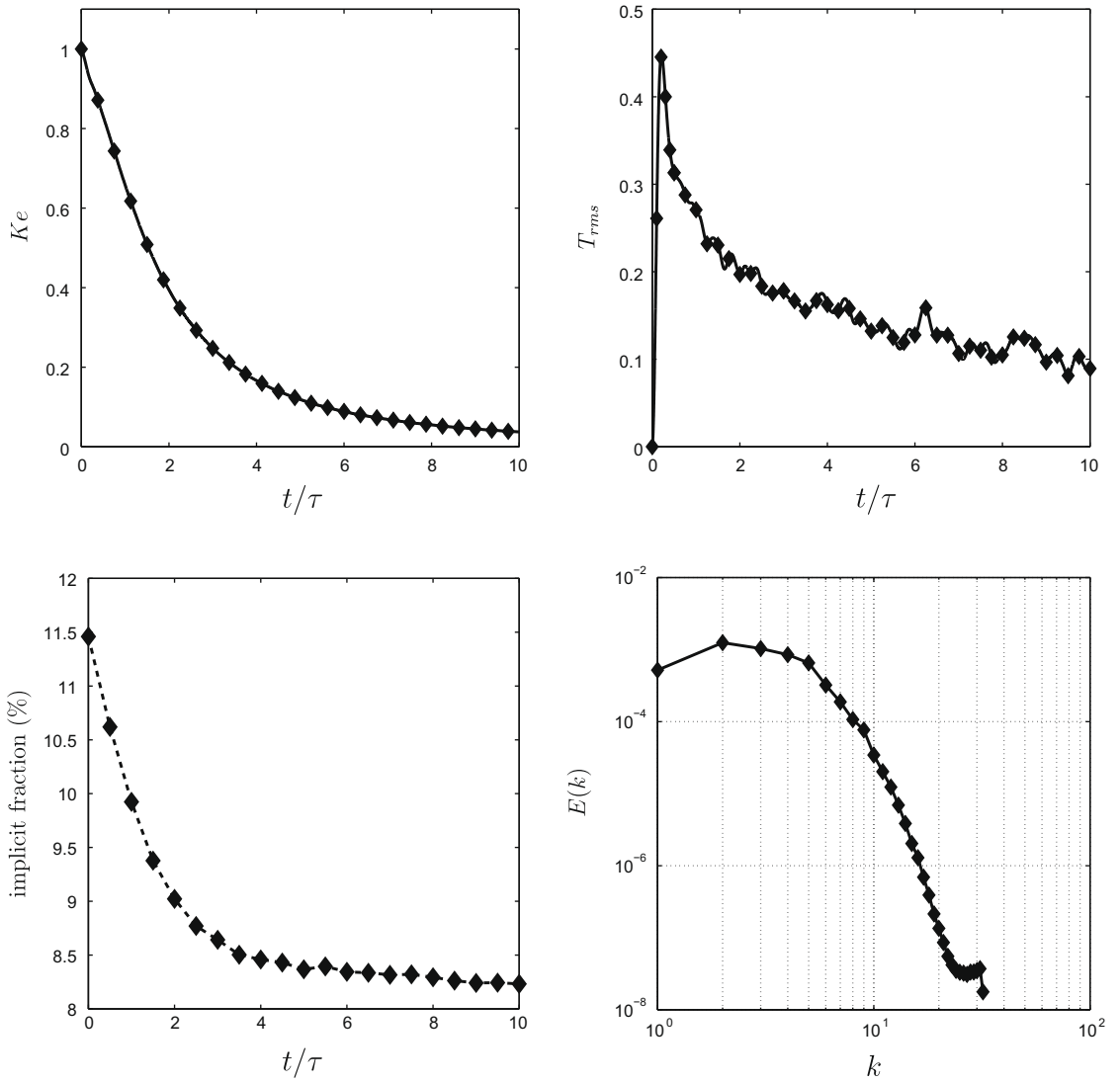

Fig. 14. Comparison between pure explicit (—) and mixed implicit–explicit (♦) time-advancement on the unstructured grid. Top-left: turbulent kinetic energy normalized by the initial value; top-right: temperature fluctuations $T_{rms}/[(\gamma-1)M_{t_0}^2]$; bottom-left: implicit fraction; bottom-right: 3D energy spectra at $t/\tau = 5$.

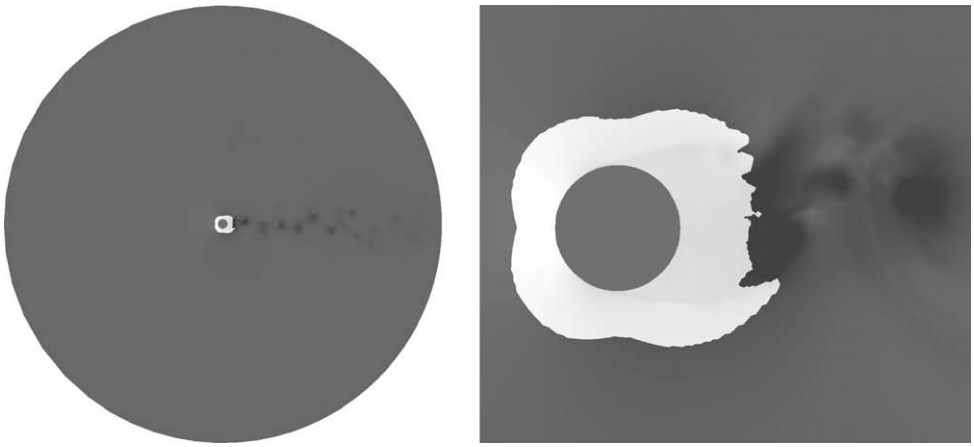

Fig. 15. Implicit/explicit mapping; Left: entire domain; right: near cylinder region. White regions are treated using implicit time-advancement.

of the subgrid stress tensor is given in Fig. 16. The white regions in the viscosity ratio contours are generated when the viscosity is clipped to zero.

In Fig. 17, the energy spectra at 2*D* and 5*D* downstream of the cylinder are compared with simulation results of Mani et al. [46] in which they used a staggered sixth-order compact Pade scheme of [43]. Due to sufficient resolution near the cylinder, the spectra are in good agreement even at high frequencies. However, as the grid becomes coarser downstream in the wake, the high-order code is able to better resolve the high wavenumber fluctuations than the present second-order scheme. In any case, the Strouhal number and the energy content up to at least five shedding frequencies are well-predicted.

In the rest of this section, we assess the performance of the RS-IMEX scheme. We begin by studying the performance of the proposed scheme as the number of grid points per processor change. The computations have been carried out on the Stanford WCR computer cluster [49], which has Intel Quad-Core Clovertown 2.33 GHz compute nodes with 1 GB RAM per processor and uses Cisco DDR infiniband interconnect. The number of grid points per processor varies from 12,500 (256 processors) to 133,000 (24 processors) for which the code reaches the limit of 1 GB memory per processor. The wall-clock time

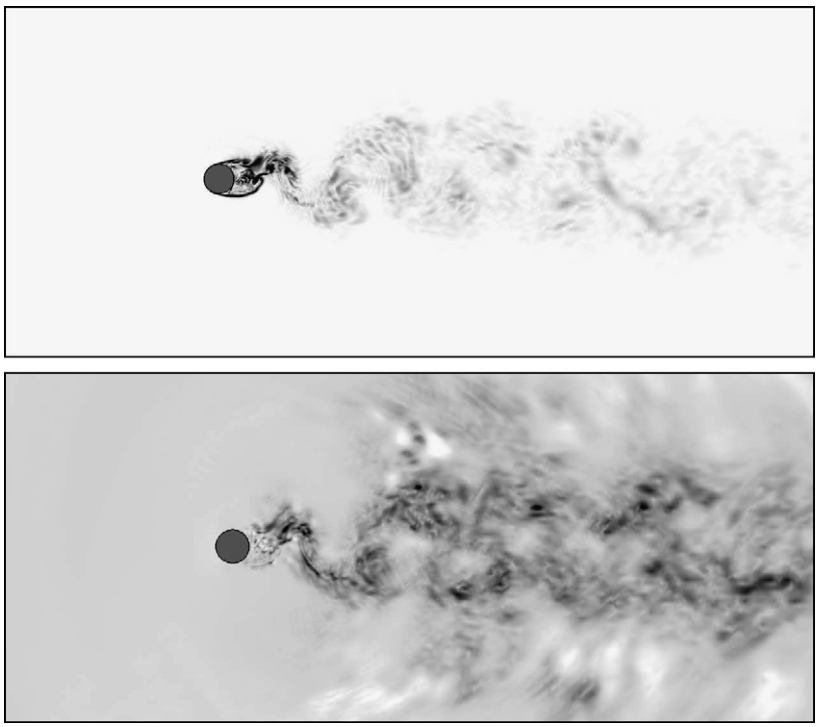

Fig. 16. Top: vorticity magnitude contours; bottom: $\frac{\nu}{\nu_m}$, ratio of total viscosity to the molecular viscosity. For both figures contours are from 0 to 6.

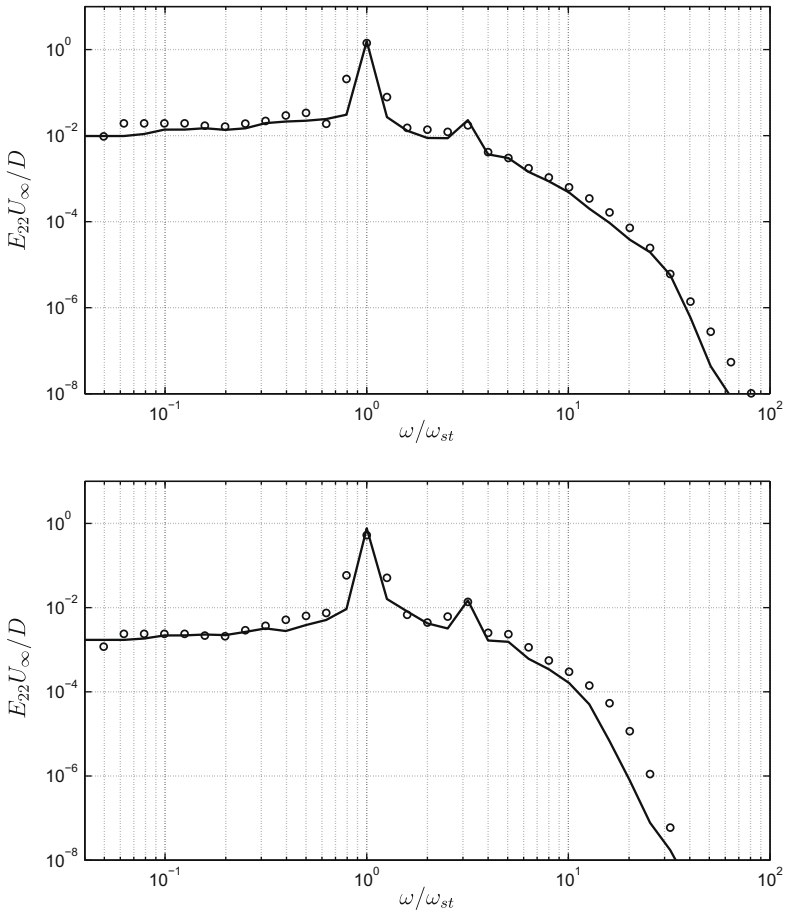

Fig. 17. Comparison of the energy spectra of the vertical velocity as a function of non-dimensional frequency at $x/D = 2$ (top) and $x/D = 5$ (bottom). —: current results; ○: Mani et al. [46]

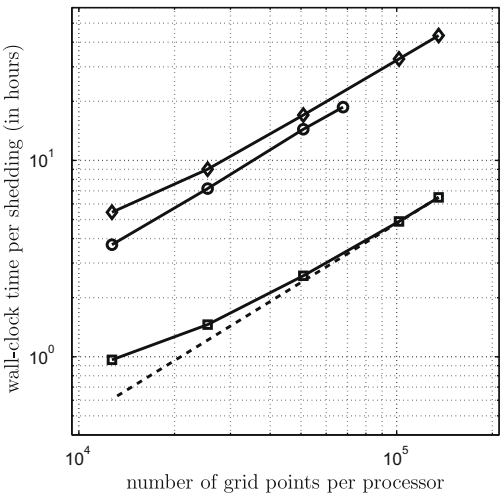

Fig. 18. Wall-clock time required to advance the flow for one vortex shedding period versus the number of grid points per processor for different methods; □: IMEX algorithm (Case 1); ○: fully implicit (Case 2); ♢: simulation with CFL ≈ 1 (Case 3); ----: ideal scaling for Case 1.

needed to advance the flow one vortex shedding period versus the number of grid points per processor is plotted in Fig. 18. Three different cases are considered: (1) RS-IMEX method with CFL ≈ 16; (2) fully implicit method with the same CFL; (3) simulation at CFL ≈ 1. Note that for Case 3 all the grid points except the ones on the cylinder wall have their fluxes advanced

explicitly in time. Again, the implicit treatment on the cylinder wall is required due to the wall penalty terms added to impose no-slip conditions (see Section 4.2).

As the number of grid points per processor decreases, the wall-clock time for the RS-IMEX scheme deviates more rapidly from the ideal scaling than in the other two cases. This deviation is mainly due to two reasons: first, the implicit solver in the RS-IMEX method requires extra communications between the processors than the explicit method; and second, the dual-constraint partitioning in the RS-IMEX scheme yields disjointed partitions thus resulting in a larger edge-cut than explicit and fully implicit methods and consequently an increase in the communication time. For example, on 64 processors, single-constraint partitioning results in an edge-cuts of 290,705, whereas it increases by 20% to 350,689 when dual-constraint partitioning is used. Nevertheless, even for a small number of grid points per processor, the RS-IMEX method results in over five times increased efficiency. Moreover, the full implicit method was stopped at 66,000 grid points per processor because it reached the limit of 1 GB memory per processor, whereas the RS-IMEX simulations can be performed with up to 140,000 grid points per processor.

Strong and weak scalabilities of the RS-IMEX approach are plotted in Fig. 19. For the strong scalability the problem size is fixed to 3.2 million grid points. The speedup is defined as

$$S(p) = \frac{T_{(p_{ref})}}{T_{(p)}}, \tag{21}$$

where $T_{(p)}$ is the wall-clock time for computing the flow for one vortex shedding using p processors and p_{ref} is the reference case. Here, the case with 24 processors is taken as the reference, i.e., $p_{ref} = 24$. From a practical point of view, most of the simulations are performed with 50,000 to 100,000 grid points per processor, for which the speedup is more than 90% of the ideal case.

The weak scaling is more difficult to study for the RS-IMEX method. By increasing the number of processors, we try to keep the number of grid points per processor constant. This implies having the same implicit mapping for problems with different sizes, which is the main reason why only two data points have been collected for this study. To have the same mapping, we take the simulation grid with 3.2 million grid points and measure the time required to advance 100 time steps on 32 processors [denoted $\widetilde{T}_{(32)}$]. Then the grid is refined homogeneously in all directions, which results in a grid that has eight times more grid points. The time-step size is split in half and the simulation is performed on 256 processors. Again, the time, $\widetilde{T}_{(256)}$, required to march 100 time steps is measured. With a half-time-step size, the implicit mapping remains almost constant. On 32 processors, on average, the implicit fraction is 27.25% and on 256 processors it is 27.15%. For this study, the efficiency

$$E(p) = \frac{\widetilde{T}_{(p_{ref})}}{\widetilde{T}_{(p)}}, \tag{22}$$

where $p_{ref} = 32$, decreased to 0.86 on 256 processors.

One should note that the performance of the RS-IMEX method is a strong function of the CFL number. At CFL numbers close to unity, the algorithm will advance all the terms explicitly in time and no gain is obtained, whereas for large time-step sizes, the method would behave as a fully implicit algorithm. We define the performance of the RS-IMEX method as

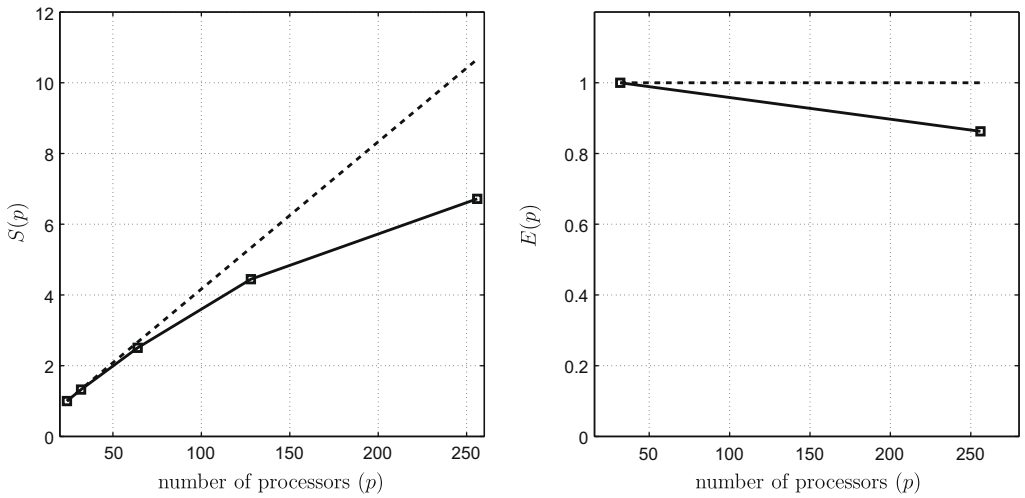

Fig. 19. Scalability study for the flow around the cylinder at CFL = 16. Left: strong scaling; right: weak scaling. For both figures, □: simulation results; ----: ideal scaling.

Table 4
Performance (ϵ) of the RS-IMEX method and implicit fractions as a function of the CFL number. The simulation is performed on 64 processors.

CFL	1	2	4	8	16	32
$\epsilon_{(CFL)}$	1.00	1.82	2.44	4.00	6.67	5.26
Implicit fraction (%)	0.31	1.69	7.38	16.8	27.5	38.9

$$\epsilon_{(CFL)} = \frac{\overline{T}_{(CFL=1)}}{\overline{T}_{(CFL)}}, \tag{23}$$

where $\overline{T}_{(CFL)}$ is the wall-clock time for computing the flow for one vortex shedding at a given CFL number. For flow around the cylinder, this parameter is calculated at six different CFL numbers and the results are given in Table 4. The performance initially increases with the CFL number and then decreases. This drop in efficiency beyond a certain CFL number is due to the increase in the cost of the linear and non-linear solvers. We note that for non-stationary solutions, the CFL will also be limited by the required temporal accuracy.

4.6. LES of a transonic turbulent jet

To assess the proposed RS-IMEX approach and its capability of performing high-fidelity large-scale numerical simulations on massively parallel computers, LES of a turbulent jet with acoustic Mach number, $Ma = U_j/c_\infty = 0.89$ and Reynolds number, $Re_D = U_j D/\nu_j = 1.2 \times 10^5$ is performed. Here, U_j is the jet exit velocity, c_∞ is the ambient speed of sound, D is the nozzle diameter, and ν_j is the jet's kinematic viscosity. A cold jet with static temperature ratio $T_j/T_\infty = 0.84$ is considered. The prescribed geometry of the nozzle is the Acoustic Reference Nozzle (ARN) [50]. The computational domain is a cylinder with radius $R = 22.5D$ and height $H = 56.5D$ ($-10D \leqslant z \leqslant 46.5D$) with the nozzle exit located at $z = 3.5D$. The domain is decomposed with 35.2 million grid points. The mesh uses hexahedral elements and is effectively axisymmetric with 128 elements in the azimuthal direction near the nozzle geometry and unstructured in the core to avoid the centerline coordinate singularity. The mesh is designed with highly anisotropic elements near the nozzle lip boundary layer (see Fig. 20).

To perform scalability study, the time-step is chosen such that CFL $\approx$ 20. The wall-normal spacing of the first element near the exit of the nozzle is $0.0001D$. Hence, the region near the nozzle restricts the time-step size thus requiring implicit time-advancement. Regions with implicit nodes are shown in Fig. 20. The present algorithm clusters the implicit grid points near the nozzle with implicit fraction and nodal implicit fractions 14% and 7/27, respectively. The strong scaling study for this case

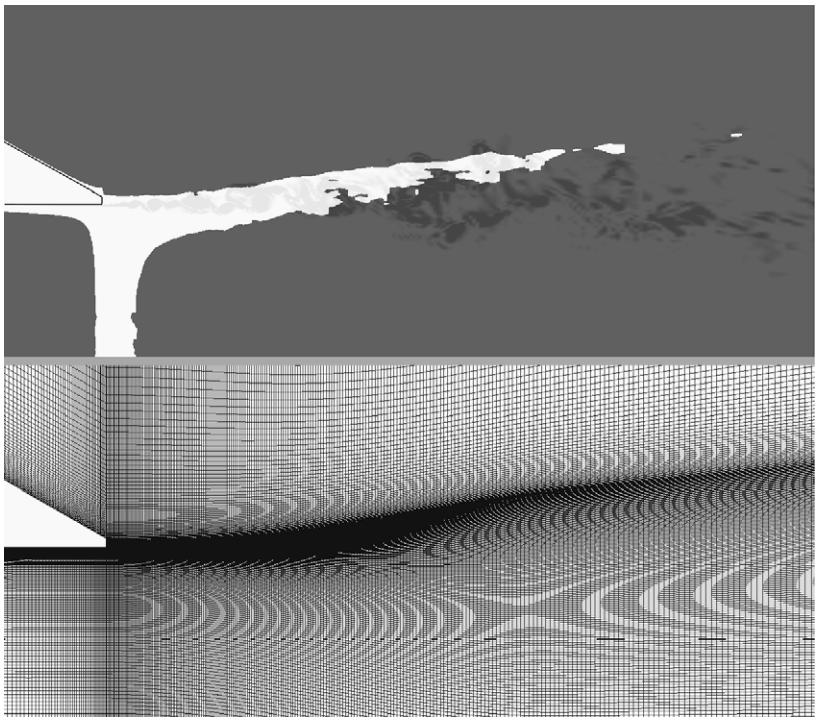

Fig. 20. Near nozzle region. Top: time-implicit mapping; bottom: grid.

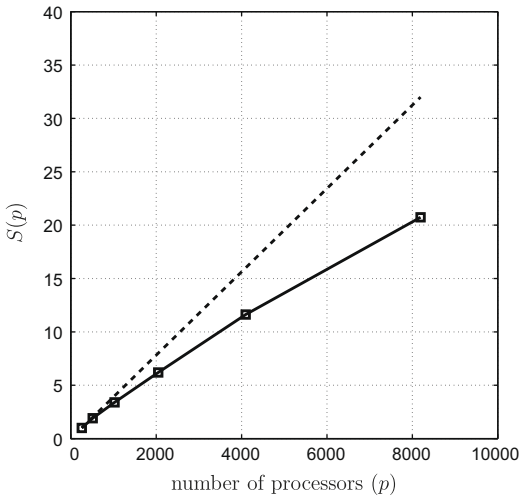

Fig. 21. Scalability study for the LES of the transonic jet. Speedup curve is with respect to the case with 256 processors. □ : simulation results; ----: ideal scaling.

is performed on Argonne Blue Gene/P IBM cluster [51] using up to 8192 core. Each core has a 850 MHz PowerPC 450 processor. The speedup is defined in the same way as in (21) with $p_{ref} = 256$ and $T_{(p)}$ being the averaged wall-clock time required to advance the equations one time-step. Fig. 21 shows the speedup as a function of the number of processors. On 256 cores, there are 137,500 grid points per processor, whereas there is only 4300 grid points per processor using 8192 cores. The efficiency is more than 75% up to 4096 cores (corresponding to 8600 grid points per processor) and it drops to 65% using 8192 cores.

5. Conclusions

We have designed an efficient numerical scheme for DNS and LES of compressible flows on unstructured grids suitable for realistic applications. For the spatial discretization, we have used a node-based SBP–SAT finite-volume method with provable stability properties on general unstructured meshes. The main contribution of this paper is the Row-Splitting algorithm that decomposes the computational problem into stiff and non-stiff parts and enables the use of an implicit–explicit Runge–Kutta temporal scheme. It was shown that the splitting did not introduce any eigenvalues with positive real parts, which would have introduced instability in the split system. For multi-dimensional problems, we also proposed a simple change of the algorithm to further reduce the size of the implicit set.

Second-order convergence of the scheme was verified for a 2D propagating vortex on prisms. Further tests for which we have validated the method were: a boundary-layer transition with TS waves; a 2D laminar flow over a cylinder on a mixed grid; DNS of homogeneous isotropic turbulence on both hexahedral and tetrahedral elements; LES of the flow over a cylinder with a turbulent wake; and LES of a transonic turbulent jet.

To balance both the computational work and the required memory on multiple processors, a dual-constraint partitioning method was employed. Scalability and efficiency studies have been performed, and it was shown that in addition to a significant memory-saving compared with a fully implicit method, a notable reduction in computer cost is achieved.

Acknowledgments

The work is supported by the Department of the Energy under the ASC program and Stanford Graduate Fellowship. The authors are grateful to Dr. Ali Mani for providing the LES data of flow over cylinder, to Ms. Taraneh Sayadi for providing the similarity solution for the laminar boundary layer and to Dr. Simon Mendez for helpful comments during the preparation of the manuscript.

References

[1] M.P. Martin, G.V. Candler, A parallel implicit method for the direct numerical simulation of wall-bounded compressible turbulence, J. Comput. Phys. 215 (2005) 153–171.
[2] G. Jothiprasad, D.J. Mavriplis, D.A. Caughey, Higher-order time integration schemes for the unsteady Navier–Stokes equations on unstructured meshes, J. Comput. Phys. 191 (2003) 542–566.
[3] Y. Hou, K. Mahesh, A robust, collocated, implicit algorithm for direct numerical simulation of compressible turbulent flows, J. Comput. Phys. 205 (2005) 205–221.
[4] C. Wall, C.D. Pierce, P. Moin, A semi-implicit method for resolution of acoustic waves in low Mach number flows, J. Comput. Phys. 181 (2002) 545–563.

[5] D.A. Knoll, D.E. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–397.
[6] J. Frank, W. Hundsdorfer, J.G. Verwer, On the stability of implicit–explicit linear multistep methods, Appl. Numer. Math. 25 (1997) 193–205.
[7] U.M. Ascher, S.J. Ruuth, R.J. Spiteri, Implicit–explicit Runge–Kutta methods for time-dependent partial differential equations, Appl. Numer. Math. 25 (1997) 151–167.
[8] C.A. Kennedy, M.H. Carpenter, Additive Runge–Kutta schemes for convection–diffusion-reaction equations, Appl. Numer. Math. 44 (2003) 139–181.
[9] P.R. Spalart, D.M. Moser, Spectral methods for the Navier–Stokes equations with one infinite and two periodic directions, J. Comput. Phys. 96 (1991) 297–324.
[10] H. Le, P. Moin, An improvement of fractional step methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 92 (1991) 369–379.
[11] X. Zhong, Additive semi-implicit Runge–Kutta methods for computing high-speed nonequilibrium reactive flows, J. Comput. Phys. 128 (1996) 19–31.
[12] Y. Morinishi, T.S. Lund, O.V. Vasilyev, P. Moin, Fully conservative higher order finite difference schemes for incompressible flow, J. Comput. Phys. 143 (1998) 90–124.
[13] I. Nompelis, T.W. Drayna, G.V. Candler, A parallel unstructured implicit solver for hypersonic reacting flow simulation, in: 17th AIAA Computational Fluid Dynamics Conference, Toronto, Canada, June 6–9, 2005.
[14] I. Men'shov, Y. Nakamura, Hybrid explicit–implicit, unconditionally stable scheme for unsteady compressible flows, AIAA J. 42 (3) (2004) 551–559.
[15] A. Kanevsky, M.H. Carpenter, D. Gottlieb, J.S. Hesthaven, Application of implicit–explicit high order Runge–Kutta methods to discontinuous-Galerkin schemes, J. Comput. Phys. 225 (2007) 1753–1781.
[16] H.O. Kreiss, G. Scherer, Finite element and finite difference methods for hyperbolic partial differential equations, in: Mathematical Aspects of Finite Elements in Partial Differential Equations, Academic Press Inc., 1974.
[17] M. Svärd, J. Nordström, A stable high-order finite difference scheme for the compressible Navier–Stokes equations: no-slip wall boundary conditions, J. Comput. Phys. 227 (2008) 4805–4824.
[18] M. Svärd, M. Carpenter, J. Nordström, A stable high-order finite difference scheme for the compressible Navier–Stokes equations, far-field boundary conditions, J. Comput. Phys. 225 (2007) 1020–1038.
[19] J. Nordström, K. Forsberg, C. Adamsson, P. Eliasson, Finite-volume methods, unstructured meshes and strict stability for hyperbolic problems, Appl. Numer. Math. 45 (2003) 453–473.
[20] M. Svärd, J. Nordström, Stability of finite-volume approximations for the Laplacian operator on quadrilateral and triangular grids, Appl. Numer. Math. 51 (2005) 101–125.
[21] M. Svärd, J. Gong, J. Nordström, Stable artificial dissipation operators for finite-volume schemes on unstructured grids, Appl. Numer. Math. 56 (2006) 1481–1490.
[22] A.E. Honein, P. Moin, Higher entropy conservation and numerical stability of compressible turbulence simulations, J. Comput. Phys. 201 (2004) 531–545.
[23] A. Jameson, Formulation of kinetic energy preserving conservative schemes for gas dynamics and direct numerical simulation of one-dimensional viscous compressible flow in a shock tube using entropy and kinetic energy preserving schemes, J. Sci. Comput. 34 (2008) 188–208.
[24] P.K. Subbareddy, G.V. Candler, A fully discrete, kinetic energy consistent finite-volume scheme for compressible flows, J. Comput. Phys. 228 (2009) 1347–1364.
[25] F. Ducros, F. Laporte, T. Soulres, V. Guinot, P. Moinat, B. Caruelle, High-order fluxes for conservative skew-symmetric-like schemes in structured meshes: application to compressible flows, J. Comput. Phys. 161 (2000) 114–139.
[26] J. Nordström, Conservative finite difference formulations, variable coefficients, energy estimates and artificial dissipation, J. Sci. Comput. 29 (2005) 375–404.
[27] F.E. Ham, K. Mattsson, G. Iaccarino, Accurate and stable finite-volume operators for unstructured flow solvers, Ann. Res. Briefs Center Turbulence Res. (2006) 243–261.
[28] K. Mattsson, M. Svärd, M. Shoeybi, Stable and accurate schemes for the compressible Navier–Stokes equations, J. Comput. Phys. 227 (2008) 2293–2316.
[29] P. Moin, Fundamentals of Engineering Numerical Analysis, Cambridge University Press, 2001.
[30] H.O. Kreiss, L. Wu, On the stability definition of difference approximations for the initial boundary value problem, Appl. Numer. Math. 12 (1993) 213–227.
[31] G.L.G. Sleijpen, H.A. van der Vorst, Hybrid bi-conjugate gradient methods for CFD problems, Comput. Fluid Dynam. Rev. (1995).
[32] H. VanDerVorst, Iterative Methods for Large Linear Systems, Cambridge University Press, 2003.
[33] G. Karypis, V. Kumar, Multilevel algorithms for multi-constraint graph partitioning, in: Proceedings of the ACM/IEEE Supercomputing, 1998.
[34] G. Karypis, K. Schloegel, V. Kumar, ParMETIS-parallel graph partitioning and sparse matrix ordering library, 2003. Available from <http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview>.
[36] J.D. Anderson Jr., Hypersonic and High Temperature Gas Dynamics, McGraw-Hill, New York, 1989.
[37] H. Fasel, U. Konzelmann, Non-parallel stability of a flat-plate boundary layer using the complete Navier–Stokes equations, J. Fluid Mech. 221 (1990) 311–347.
[38] M. Gaster, On the effects of boundary-layer growth on flow stability, J. Fluid Mech. 66 (3) (1974) 465–480.
[39] S. Nagarajan, J.H. Ferziger, S.K. Lele, Leading Edge Effect in Bypass Transition, Rep. Num TF-90, Stanford University, 2004.
[40] K. Kwon, H. Choi, Control of laminar vortex shedding behind a circular cylinder using splitter plates, Phys. Fluids 8 (2) (1996) 479–486.
[41] A.G. Kravchenko, P. Moin, B-Spline Methods and Zonal Grids for Numerical Simulations of Turbulent Flows, Rep. Num TF-73, Stanford University, 1998.
[42] U. Fey, M. König, H. Eckelmann, A new Strouhal–Reynolds-number relationship for the circular cylinder in the range $47 < Re < 2 \times 10^5$, Phys. Fluids 10 (7) (1998) 1547–1549.
[43] S. Nagarajan, S.K. Lele, J.H. Ferziger, A robust high order compact method for large eddy simulation, J. Comput. Phys. 191 (2003) 392–419.
[44] G.A. Blaisdell, N. Mansour, W. Reynolds, Numerical Simulation of Compressible Homogenous Turbulence, Rep. Num TF-50, Stanford University, 1991.
[45] A.G. Kravchenko, P. Moin, Numerical studies of flow over a circular cylinder at $Re_D = 3900$, Phys. Fluids 12 (2) (2000) 403–417.
[46] A. Mani, P. Moin, M. Wang, Computational study of optical distortions by separated shear layers and turbulent wakes, J. Fluid Mech. 625 (2009) 273–298.
[47] P. Moin, K. Squires, W. Cabot, S. Lee, A dynamic subgrid-scale model for compressible turbulence and scalar transport, Phys. Fluids 11 (1991) 2746–2757.
[48] D.K. Lilly, A proposed modification of the Germano subgrid-scale closure method, Phys. Fluids A 4 (1992) 633–635.
[49] Stanford University, High-Performance Computer Center, WCR cluster, <http://hpcc.stanford.edu/clusters/wcr.html>.
[50] J. Bridges, C.A. Brown, Validation of the Small Hot Jet Acoustic Rig for Aeroacoustic Research, in: AIAA Paper 2005-2846, 11th AIAA/CEAS Aeroacoustics Conference, 2005.
[51] Argonne National Laboratory, Argonne Leadership Computing Facility (ALCF), <http://wiki.alcf.anl.gov/index.php/Main_Page>.