GePUP: Generic Projection and Unconstrained PPE for Fourth-Order Solutions of the Incompressible Navier–Stokes Equations with No-Slip Boundary Conditions

Qinghai Zhang

Journal of Scientific Computing

ISSN 0885-7474
Volume 67
Number 3

J Sci Comput (2016) 67:1134-1180
DOI 10.1007/s10915-015-0122-4

Springer

J Sci Comput (2016) 67:1134–1180
DOI 10.1007/s10915-015-0122-4

GePUP: Generic Projection and Unconstrained PPE for Fourth-Order Solutions of the Incompressible Navier–Stokes Equations with No-Slip Boundary Conditions

Qinghai Zhang[1]

Received: 20 June 2014 / Revised: 22 September 2015 / Accepted: 10 October 2015 /
Published online: 18 October 2015

Abstract A generic projection maps one vector to another such that their difference is a gradient field and the projected vector does not have to be solenoidal. Via a commutator of Laplacian and the generic projection, the projected velocity is formulated as the sole evolutionary variable with the incompressibility constraint enforced by a pressure Poisson equation so that the dissipation of velocity divergence is governed by a heat equation. Different from previous projection methods, the GePUP formulation treats the time integrator as a black box. This prominent advantage is illustrated by straightforward formations of a semi-implicit time-stepping scheme and another explicit time-stepping scheme. Apart from its stability, the GePUP schemes have an optimal efficiency in that within each time step the solution is advanced by solving a sequence of linear systems with geometric multigrid. A key component of the GePUP schemes is a fourth-order discrete projection for no-penetration domains. Results of numerical tests in two and three dimensions demonstrate that the GePUP schemes are fourth-order accurate both in time and in space. To facilitate efficiency comparison to other methods, a simple formula is introduced. Systematic arguments and timing results show that the GePUP schemes could be vastly superior over lower-order methods in terms of efficiency and accuracy. In some cases, the GePUP schemes running on the author's personal desktop would be faster than a second-order method running on the fastest supercomputer in the world! This paper contains enough details so that one can reproduce the numerical results by following the exposition.

Keywords Incompressible Navier–Stokes equations · No-slip boundary conditions · Generic projection · Unconstrained pressure Poisson equation · Lid-driven cavity · Laplace–Leray commutator

In memory of my mentor, Madam Yin, Ping.

✉ Qinghai Zhang
tsinghai@gmail.com

[1] Department of Mathematics, University of Utah, 155 S. 1400 E. Rm233, Salt Lake City, UT 84112, USA

Mathematics Subject Classification 76D05 · 65M08 · 65L06 · 35Q30

1 Introduction

At the center of modern mathematical physics are the incompressible Navier–Stokes equations (INSE),

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} + \mathbf{g} - \nabla p + \nu \Delta \mathbf{u}, \tag{1a}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{1b}$$

where t is time, $\mathbf{x} \in \mathbb{R}^{\mathrm{D}}$ ($\mathrm{D} = 2, 3$) the spatial location, $\mathbf{g}$ the external forcing term, p the pressure, and $\mathbf{u}$ the velocity. The dynamic viscosity ν is usually a constant.

At the presence of no-slip boundaries, flows with moderate or high Reynolds number tend to develop structures of multiple length scales and time scales. This poses a major challenge of computational efficiency: the numerical method must have sufficient accuracy to capture features of all relevant scales. For first-order and second-order methods, solutions are cheap to compute, but computational resources may be rapidly exhausted. For most methods that employ explicit time integration, the discrete system becomes stiff for large viscosity, and time steps might be unnecessarily constrained to very small sizes. In addition, the approach of treating the INSE as an index-2 differential algebraic system with explicit Runge–Kutta methods often suffers from large order reductions in pressure [45]. Fully implicit methods rely on expensive solvers that must be able to simultaneously handle both the non-linearity and the divergence-free constraint [43]. Spectral methods have been successful in achieving high-resolution simulations [25,40], but come with high computational cost from dense linear systems, and sometimes require additional stabilization and conservation procedures [5,31,41]. These remarks motivate high-order methods that can handle a wide range of flow regimes with an *optimal* efficiency: the computational cost of advancing the solution for each time step is *linear* with respect to the number of control volumes in the domain.

Another major challenge of numerically solving the INSE is how to enforce the incompressibility constraint (1b). To this end, one can discretize the INSE into a saddle point system to solve for the velocity and the pressure simultaneously; see [18] for an example. In comparison, projection methods decouple the solutions of the two primary variables by first computing an auxiliary velocity field and then projecting the auxiliary velocity onto the divergence-free space to enforce (1b). Consequently, the INSE is advanced by solving a sequence of boundary value problems (BVPs) either on the velocity or on the pressure. The number of unknowns of each linear system is thus much smaller than that in the saddle point approach. This efficiency advantage is probably one of the reasons for the popularity of projection methods. The original projection method by Chorin [13] is first-order accurate, and many successful second-order projection methods have been proposed, e.g. [6,15,28,42]; see [11] and [20] for two excellent overviews.

In the saddle-point formulation, the pressure is coupled to the velocity according to the particularities of a time-marching scheme. As such, it is difficult to achieve high-order accuracy in time since the linear systems resulting from the internal details of a time-marching scheme may become unacceptably complicated. As for second-order projection methods, although the pressure and the velocity are conceptually decoupled, their boundary conditions are still coupled in time according to the details of temporal integration. This temporal cou-

pling of the boundary conditions tends to generate numerical boundary layers and hinders the generalization of these methods to higher-order accuracy.

During recent years, it has become increasingly clear that the pressure Poisson equation (PPE) formulation [17,22,24,29,33,34,48] offers a straightforward, accurate, and efficient way in obtaining high-order solutions for the INSE. In the original PPE formulation, (1b) and the divergence of (1a) yield

$$\Delta p = \nabla \cdot (\mathbf{g} - (\mathbf{u} \cdot \nabla)\mathbf{u} + \nu \Delta \mathbf{u}), \tag{2}$$

which replaces the incompressible constraint. If $\nabla \cdot \mathbf{u} = 0$ is further imposed as an boundary condition, (1a) and (2) are equivalent to the original INSE [17].

From the viewpoint of (1a) and (2), the velocity is the only evolution variable with the pressure regarded as its implicit function. The coupling of the pressure and the velocity through (2) is only *in space*, which is also true for their boundary conditions. Consequently, the PPE formulation admits a "black-box" usage of time-marching schemes in that the internal details of the time-marching scheme are not needed. As a large benefit, high-order solutions in time can be obtained from "samples" of velocity at time instances specified by the signature of a time-marching scheme.

The original PPE formulation is by no means perfect. For example, the derivation of (2) assumes that the constraint (1b) holds for the entire evolution of the velocity. However, in practice the divergence-free constraint can only be enforced for a limited number of times, which undermines the validity of (2) as a governing equation. Meanwhile, (1a) and (2) imply that the dissipation of velocity divergence is degenerate, i.e. $\frac{\partial \nabla \cdot \mathbf{u}}{\partial t} = 0$; this invites numerical oscillations and makes it hard to analyze the influence of a divergent velocity upon the solution.

Recently, Liu et al. [33] proposed a novel PPE formulation for no-slip domains via a commutator estimate of the Laplacian and the Leray–Helmholtz projection. In this *unconstrained PPE (UPPE) formulation*, the pressure is strategically chosen so that the dissipation of velocity divergence is governed by a heat equation, leading to a more robust control of velocity divergence than that in the original PPE formulation. The well-posedness and strong stability of explicit treatment of pressure gradient in time are shown both by theoretical analysis [33] and numerical experiments with finite elements [34].

A major difficulty in applying the UPPE formulation to finite-volume methods on structured grids is the absence of a fourth- or higher-order discrete projection $\mathbf{P}$ that satisfies the properties of a Leray–Helmholtz projection. More precisely, it is difficult for $\mathbf{P}$ to satisfy any of the following:

$$\mathbf{P}^2 = \mathbf{P}, \quad \mathbf{DP}\langle \mathbf{u} \rangle = 0, \quad \mathbf{PG}\langle \phi \rangle = \mathbf{0}, \tag{3}$$

where $\langle \mathbf{u} \rangle$ and $\langle \phi \rangle$ are vectors of cell averaged vectors and scalars, respectively, and $\mathbf{D}$ and $\mathbf{G}$ are respectively the discrete divergence and discrete gradient. Note that this difficulty also holds on staggered grids because of the non-periodicity of the domain and the fourth-order accuracy. Rigorously speaking, it is even inappropriate to employ in the UPPE formulation a discrete projection $\tilde{\mathbf{P}}$ that does not satisfy (3), because the derivation of the governing equations of UPPE is based on the Leray–Helmholtz projection. More importantly, it is not clear how numerical stability would be affected by the error in approximating the Leray–Helmholtz projection with $\tilde{\mathbf{P}}$.

The author resolves these difficulties by proposing GePUP, a new formulation of the INSE via a Generic Projection and an Unconstrained pressure Poisson equation. A generic projection $\mathcal{P}$ is a decomposition of one vector field $\mathbf{v}^*$ into another vector field $\mathbf{v}$ and a

gradient field $\nabla\phi$, i.e. $\mathcal{P}\mathbf{v}^* = \mathbf{v} := \mathbf{v}^* - \nabla\phi$, meanwhile each of the following may or may not hold,

$$\mathcal{P}^2 = \mathcal{P}, \quad \nabla \cdot \mathcal{P}\mathbf{u} = 0, \quad \mathcal{P}\nabla\phi = \mathbf{0}. \tag{4}$$

After deriving a commutator for this generic projection and the Laplacian, the projected velocity is reformulated as the only evolutionary variable, with the incompressibility constraint enforced by a pressure Poisson equation. Similar to UPPE, this GePUP formulation also has the advantage that the dissipation of velocity divergence is governed by a heat equation. However, GePUP is more general in that (i) the UPPE formulation can be shown as a special case of GePUP, and (ii) the GePUP formulation provides an additional level of flexibility in designing numerical algorithms: the discrete projection does not have to satisfy any conditions in (3). Furthermore, it is clearly visible how numerical accuracy and stability are influenced by a generic projection for which (4) does not hold.

Another major contribution of this work is a fourth-order accurate discrete projection for rectangular no-penetration domains as an extension of the one for periodic domains [57]. It is then used to team up with the GePUP formulation and method-of-lines to construct two groups of time-marching schemes for solving the INSE with fourth-order accuracy both in time and in space. In GePUP-IMEX, the potentially stiff diffusion term is treated implicitly while the other non-stiff terms are treated explicitly to efficiently handle a wide range of incompressible flow regimes. The other time-marching scheme GePUP-ERK adopts an explicit Runge–Kutta (ERK) method and is thus more efficient than GePUP-IMEX for flows with high-Reynolds number. As an advantage over other ERK methods [45] that treats the INSE as an index-2 differential algebraic system, the GePUP-ERK scheme retain fourth-order accuracy both for the velocity and for the pressure. Both GePUP-IMEX and GePUP-ERK have asymptotically optimal efficiency in that within each time step the solution is advanced by explicitly evaluating discrete operators and solving a sequence of linear systems via classic geometric multigrid methods.

While high-order methods are known to be superior to second-order methods in many applications [2,23,54], other researchers report disappointing findings [21,30,49] about the efficiency of a certain group of fourth-order methods. However, the fourth-order methods used in [21,30,49] are special in that their fourth-order accuracy is only in space whereas the temporal accuracy is of the second order; this probably explains the disappointment for unsteady flows since the solution errors are dominated by the second-order temporal errors. On the other hand, the disappointing results reported in [21,30,49] suggest that the fourth-order accuracy in time is also needed to fully reap the benefits of *truly* fourth-order methods.

Although there exist many comparisons of fourth-order methods to second-order methods in the literature, most (if not all) of them lack a coherent framework as they are performed on a case-by-case basis for isolated accuracy requirements. A relatively minor contribution of this work is the systematic arguments and a simple formula that can be used to compare the efficiency of different methods for a wide range of accuracy requirements. As the main conclusion, a truly fourth-order accurate method could be vastly superior to second-order methods, especially when derivatives of the primary variables are concerned.

The rest of this paper is organized as follows. The GePUP formulation is proposed in Sect. 2. In Sect. 3, various discrete operators that act on cell-averaged values are reviewed within the finite-volume framework. In particular, a fourth-order discrete projection is constructed and examined in Sect. 4. Approximating the continuous operators in the governing equations with these discrete operators leads to a system of ordinary differential equations (ODEs), for which the GePUP-IMEX and GePUP-ERK time-marching schemes are proposed in

Sect. 5. Benchmark problems are numerically solved in Sect. 6 to confirm the fourth-order convergence rates of the GePUP schemes and their capability of capturing the true physics. The superior accuracy and efficiency of the GePUP schemes over some second-order methods [6,35] are demonstrated in Sect. 7 by systematic arguments and timing results. In the same section, a simple formula is also proposed for efficiency comparison of methods with different convergence rates. Finally, Sect. 8 concludes this paper with some future research prospects.

2 Formulation

On periodic domains, the Laplacian commutes with the Leray–Helmholtz projection, and the formulation of the INSE as the evolution of velocity can be achieved by applying the projection to the momentum equation [57]. In comparison, the Laplacian and the Leray–Helmholtz projection do not commute for domains with no-slip conditions. Consequently, the INSE in this case is much more difficult to deal with.

In Sects. 2.1 and 2.2, we first review some fundamental results on the Leray–Helmholtz projection and the Laplace–Leray commutator. Then a generic projection is defined and its corresponding Laplace–Leray commutator is determined in Sects. 2.3 and 2.4. Finally in Sect. 2.5, the author proposes the GePUP formulation and relates it to the unconstrained PPE formulation of Liu et al. [33].

2.1 The Leray–Helmholtz Projection $\mathscr{P}$

Helmholtz's theorem states that a sufficiently continuous vector field $\mathbf{v}^*$ on a bounded domain Ω in $\mathbb{R}^D$ can be uniquely decomposed into a divergence-free part $\mathbf{v}$ and a curl-free part $\nabla\phi$:

$$\mathbf{v}^* = \mathbf{v} + \nabla\phi, \tag{5a}$$

$$\nabla\cdot\mathbf{v} = 0, \quad \nabla\times\nabla\phi = \mathbf{0}. \tag{5b}$$

This is achieved by solving a BVP with a Neumann boundary condition

$$\Delta\phi = \nabla\cdot\mathbf{v}^* \qquad \text{in } \Omega, \tag{6a}$$

$$\mathbf{n}\cdot\nabla\phi = \mathbf{n}\cdot(\mathbf{v}^*-\mathbf{v}) \quad \text{on } \partial\Omega, \tag{6b}$$

where $\mathbf{n}$ denotes the outward normal of the domain boundary $\partial\Omega$. Gauss theorem and $\nabla\cdot\mathbf{v}=0$ yield $\int_\Omega \nabla\cdot\mathbf{v} = \oint_{\partial\Omega}\mathbf{n}\cdot\mathbf{v} = 0$, thus the solvability of (6) follows. Furthermore, for domains with the no-penetration boundary condition $\mathbf{u}\cdot\mathbf{n}=0$, the chain rule yields

$$\int_\Omega \mathbf{v}\cdot\nabla\phi = \int_\Omega \nabla\cdot(\mathbf{v}\phi) - \int_\Omega \phi\nabla\cdot\mathbf{v} = \oint_{\partial\Omega}\phi\mathbf{n}\cdot\mathbf{v} = 0, \tag{7}$$

where the second step follows from $\nabla\cdot\mathbf{v}=0$ and Gauss theorem. In other words, the L^2-orthogonality of the divergence-free part $\mathbf{v}$ to all gradient fields[1] is equivalent to the no-penetration condition $\mathbf{n}\cdot\mathbf{v}=0$ on $\partial\Omega$, in which case (6b) simplifies to

$$\mathbf{n}\cdot\nabla\phi = \mathbf{n}\cdot\mathbf{v}^*. \tag{8}$$

Equations (6a) and (8) uniquely define the scalar ϕ on no-penetration domains.

[1] For the INSE, a numerical method benefits from this requirement in that the accuracy of the computed velocity is largely decoupled from that of the computed pressure gradient.

A *projection* $\mathscr{P}$ is a linear operator on a vector space such that $\mathscr{P}^2 = \mathscr{P}$. The Helmholtz decomposition furnishes the Leray–Helmholtz projection $\mathscr{P}$ as

$$\mathscr{P}\mathbf{v}^* := \mathbf{v} = \mathbf{v}^* - \nabla\phi, \tag{9}$$

with ϕ uniquely determined by (6) up to an additive constant. It follows from this definition and (5) that

$$\nabla \cdot \mathscr{P}\mathbf{v}^* = 0, \tag{10a}$$

$$\mathscr{P}\nabla\phi = \mathbf{0}. \tag{10b}$$

2.2 The Laplace–Leray Commutator $[\nabla, \mathscr{P}]$

The Laplace–Leray commutator can be derived by exploiting the fact that the Laplacian commutes with both the gradient and the divergence. Define the gradient-divergence commutator

$$\mathcal{B} = [\nabla, \nabla\cdot] := \Delta - \nabla\nabla\cdot, \tag{11}$$

and we have

$$\nabla \cdot \mathcal{B} = 0, \tag{12a}$$

$$\Delta\mathscr{P} = \mathcal{B}, \tag{12b}$$

where (12a) follows from (11) and the commutativity of Δ and $\nabla\cdot$, (12b) follows from (9), the commutativity of Δ and ∇, (6a), and (11):

$$\Delta\mathscr{P}\mathbf{v}^* = \Delta(\mathbf{v}^* - \nabla\phi) = \Delta\mathbf{v}^* - \nabla\Delta\phi = \Delta\mathbf{v}^* - \nabla\nabla\cdot\mathbf{v}^* = \mathcal{B}\mathbf{v}^*. \tag{13}$$

Apply $\mathscr{P}$ to the first and third terms in the above derivation, use (10b), and we have

$$\mathscr{P}\Delta\mathscr{P}\mathbf{v}^* = \mathscr{P}\Delta\mathbf{v}^*. \tag{14}$$

The commutator is thus

$$[\Delta, \mathscr{P}] := \Delta\mathscr{P} - \mathscr{P}\Delta = (\mathcal{I} - \mathscr{P})\Delta\mathscr{P} = (\mathcal{I} - \mathscr{P})\mathcal{B}, \tag{15}$$

where $\mathcal{I}$ denotes the identity operator. By (9), $(\mathcal{I} - \mathscr{P})\mathbf{v}^* = \nabla\phi$. Hence the action of the commutator on $\mathbf{v}^*$ results in the gradient of some scalar field. In the case of $\mathbf{v}^*$ being the velocity $\mathbf{u}$ in (1), the scalar is known as the *Stokes pressure*:

$$\nabla p_s := (\Delta\mathscr{P} - \mathscr{P}\Delta)\mathbf{u}. \tag{16}$$

It follows from (15) and (16) that

$$\mathcal{B}\mathbf{u} = \mathscr{P}\mathcal{B}\mathbf{u} + \nabla p_s. \tag{17}$$

Then (10a) and (12a) yield $\Delta p_s = 0$, i.e. the Stokes pressure is harmonic. Interestingly, the vector ∇p_s lies in the intersection of the divergence-free subspace and the curl-free subspace.

Define another scalar p_c as

$$\nabla p_c := (\mathcal{I} - \mathscr{P})\,(\mathbf{g} - \mathbf{u}\cdot\nabla\mathbf{u})\,. \tag{18}$$

Apply the Leray–Helmholtz projection to (1a), use the commutator (15), invoke the definitions (16) and (18), and we have

$$\nabla p = \nabla p_c + \nu\nabla p_s. \tag{19}$$

The pressure gradient in the INSE consists of two parts: ∇p_c balances the divergence of the forcing term and the nonlinear convection term while ∇p_s accounts for the non-commutativity of the Laplacian and Leray–Helmholtz projection. In the two limiting cases of $\nu \to 0$ and $\nu \to +\infty$, the pressure gradient is respectively dominated by ∇p_c and $\nu \nabla p_s$. Consequently, the nature of the INSE is very different in these two cases.

2.3 The Generic Projection $\mathcal{P}$

A *generic projection* is a linear operator $\mathcal{P}$ on a vector space such that the difference of the two vectors is a gradient field,

$$\mathcal{P}\mathbf{v}^* = \mathbf{v} := \mathbf{v}^* - \nabla\phi, \tag{20}$$

where $\nabla \cdot \mathbf{v} = 0$ may or may not hold. Clearly, the Leray–Helmholtz projection is a special case of the generic projection.

2.4 $[\Delta, \mathcal{P}]$: The Commutator of the Laplacian and the Generic Projection

Define a gradient-divergence-projection commutator as

$$[\nabla\nabla\cdot, \mathcal{P}] := \nabla\nabla \cdot \mathcal{P} - \mathcal{P}\nabla\nabla\cdot, \tag{21}$$

where the right-hand side (RHS) cannot be simplified because in general none of the properties of the Leray–Helmholtz projection in (10) holds for the generic projection.

It follows from (20) and (11) that

$$\Delta\mathcal{P}\mathbf{v}^* = \Delta(\mathbf{v}^* - \nabla\phi) = \Delta\mathbf{v}^* - \nabla\nabla\cdot\nabla\phi = \Delta\mathbf{v}^* - \nabla\nabla\cdot(\mathbf{v}^* - \mathbf{v}) = \mathcal{B}\mathbf{v}^* + \nabla\nabla\cdot\mathcal{P}\mathbf{v}^*. \tag{22}$$

The first and last expressions in (22) imply

$$\Delta\mathcal{P} = \mathcal{B} + \nabla\nabla\cdot\mathcal{P}. \tag{23}$$

Applying $\mathcal{P}$ to the third and the last expressions in (22), we have

$$\mathcal{P}\Delta - \mathcal{P}\nabla\nabla\cdot(\mathcal{I} - \mathcal{P}) = \mathcal{P}\mathcal{B} + \mathcal{P}\nabla\nabla\cdot\mathcal{P},$$

which simplifies to

$$\mathcal{P}\Delta = \mathcal{P}\mathcal{B} + \mathcal{P}\nabla\nabla\cdot. \tag{24}$$

Then it follows from (23), (24), and (21) that the commutator of the Laplacian and the generic projection is

$$[\Delta, \mathcal{P}] := \Delta\mathcal{P} - \mathcal{P}\Delta = (\mathcal{I} - \mathcal{P})\mathcal{B} + [\nabla\nabla\cdot, \mathcal{P}]. \tag{25}$$

Clearly $[\nabla\nabla\cdot, \mathcal{P}] \equiv \mathbf{0}$ if $\mathcal{P}$ satisfies (10), in which case (25) reduces to (15). In other words, (25) is more general than the Laplace–Leray commutator (15).

2.5 GePUP: Reformulating INSE Via Generic Projection and Unconstrained PPE

The derivation of GePUP makes use of Eulerian accelerations defined as

$$\mathbf{a} := \frac{\partial \mathbf{u}}{\partial t}, \quad \mathbf{a}^* := -\mathbf{u}\cdot\nabla\mathbf{u} + \mathbf{g} + \nu\Delta\mathbf{u}. \tag{26}$$

Applying a generic projection $\mathcal{P}$ to the momentum equation (1a) and using the commutators (25) and (21) yield

$$\begin{aligned}
&\frac{\partial \mathcal{P}\mathbf{u}}{\partial t} = \mathcal{P}\left(\mathbf{g} - \mathbf{u}\cdot\nabla\mathbf{u} - \nabla p\right) + \nu\Delta\mathcal{P}\mathbf{u} - \nu(\mathcal{I}-\mathcal{P})\mathcal{B}\mathbf{u} - \nu[\nabla\nabla\cdot, \mathcal{P}]\mathbf{u}\\
&\Rightarrow \frac{\partial \mathcal{P}\mathbf{u}}{\partial t} = \mathcal{P}\left(\mathbf{g} - \mathbf{u}\cdot\nabla\mathbf{u} + \nu\mathcal{B}\mathbf{u} + \nu\nabla\nabla\cdot\mathbf{u} - \nabla p\right) - \nu\mathcal{B}\mathbf{u} + \nu\Delta\mathcal{P}\mathbf{u} - \nu\nabla\nabla\cdot\mathcal{P}\mathbf{u}\\
&\Rightarrow \frac{\partial \mathcal{P}\mathbf{u}}{\partial t} = \mathcal{P}\left(\mathbf{a}^* - \nabla p\right) - \nu\mathcal{B}\mathbf{u} + \nu\Delta\mathcal{P}\mathbf{u} - \nu\nabla\nabla\cdot\mathcal{P}\mathbf{u} \qquad (27)\\
&\Rightarrow \frac{\partial \mathcal{P}\mathbf{u}}{\partial t} = \mathbf{g} - \mathbf{u}\cdot\nabla\mathbf{u} - \nabla q + \nu\Delta\mathcal{P}\mathbf{u} - \nu\nabla\nabla\cdot\mathcal{P}\mathbf{u}, \qquad (28)
\end{aligned}$$

where (27) follows from (11) and (26), and the gradient ∇q in (28) is defined as

$$\nabla q := \mathbf{a}^* - \mathcal{P}\mathbf{a} - \nu\nabla\nabla\cdot\mathbf{u}. \qquad (29)$$

The RHS of (29) is indeed a gradient field because of the identity $\mathbf{a}^* = \mathbf{a} + \nabla p$ and the definition of the generic projection (20). The scalar q is uniquely determined up to an additive constant by solving the Neumann BVP

$$\Delta q = \nabla\cdot(\mathbf{g} - \mathbf{u}\cdot\nabla\mathbf{u}) - \nabla\cdot\mathcal{P}\mathbf{a} \quad \text{in } \Omega, \qquad (30a)$$

$$\mathbf{n}\cdot\nabla q = \mathbf{n}\cdot(\mathbf{a}^* - \nu\nabla\nabla\cdot\mathbf{u}) - \mathbf{n}\cdot\mathcal{P}\mathbf{a} \quad \text{on } \partial\Omega, \qquad (30b)$$

where (30b) follows from the normal component of (29), and (30a) follows from the divergence of (29) and the fact that divergence and Laplacian commute.

If the domain Ω is bounded, the generic projection is further required to preserve the boundary conditions of the velocity, i.e.

$$\mathcal{P}\mathbf{u} = \mathbf{u} \quad \text{on } \partial\Omega. \qquad (31)$$

The equations (28), (30), and (31) are collectively named as *the GePUP formulation of the INSE*.

2.5.1 Governing Equations for Domains with No-Slip Conditions

The no-slip conditions admit two simplifications of (30b).

- By (26), (31), and the commutativity of time derivative and generic projection, we have $\mathbf{n}\cdot\mathcal{P}\mathbf{a} = \frac{\partial u_n}{\partial t}$. The no-penetration condition $\mathbf{u}\cdot\mathbf{n} = 0$ then implies $\mathbf{n}\cdot\mathcal{P}\mathbf{a} = 0$ on $\partial\Omega$. Hence the term $\mathbf{n}\cdot\mathcal{P}\mathbf{a}$ can be dropped in (30b).
- On $\partial\Omega$, $\mathbf{u} = \mathbf{0}$ implies $\mathbf{u}\cdot\nabla\mathbf{u} = \mathbf{0}$. By (26), the term $\mathbf{a}^*$ in (30b) can be replaced by $\mathbf{g} + \nu\Delta\mathbf{u}$.

Denote the projected velocity as

$$\mathbf{w} := \mathcal{P}\mathbf{u}, \qquad (32)$$

assume two conditions on the divergence of the projected velocity,

$$\nabla\nabla\cdot\mathbf{w} = \mathbf{0}, \qquad (33a)$$

$$\frac{\partial \nabla\cdot\mathbf{w}}{\partial t} = 0, \qquad (33b)$$

and we arrive at the *GePUP formulation of the INSE with no-slip conditions*:

$$\frac{\partial \mathbf{w}}{\partial t} = \mathbf{g} - \mathbf{u}\cdot\nabla\mathbf{u} - \nabla q + \nu\Delta\mathbf{w} \ \text{ in } \ \Omega, \tag{34a}$$

$$\mathbf{w} = \mathbf{u} = \mathbf{0} \ \text{ on } \ \partial\Omega, \tag{34b}$$

$$\Delta q = \nabla\cdot(\mathbf{g} - \mathbf{u}\cdot\nabla\mathbf{u}), \ \text{ in } \ \Omega, \tag{34c}$$

$$\mathbf{n}\cdot\nabla q = \mathbf{n}\cdot(\mathbf{g} + \nu\Delta\mathbf{u} - \nu\nabla\nabla\cdot\mathbf{u}) \ \text{ on } \ \partial\Omega, \tag{34d}$$

where (34a) follows from (28) and (33a), (34b) from (32), (31), and no-slip conditions, (34c) from (30a) and (33b), and (34d) from (30b) and the aforementioned simplifications.

In (34), the projected velocity $\mathbf{w}$ is the sole evolutionary variable, with the velocity $\mathbf{u}$ and the pressure q considered as functions of $\mathbf{w}$. Indeed, a main goal of GePUP is to reformulate the INSE into the evolution of a single variable so that an ODE time integrator can be employed in a "black-box" manner. For this purpose, the functional form of $\mathbf{u}$ in terms of $\mathbf{w}$ must be specified. In this work, $\mathbf{u}$ is specified as

$$\mathbf{u} := \mathfrak{F}(\mathbf{w}) = \mathscr{P}\mathbf{w}. \tag{35}$$

The choice of the projected velocity $\mathbf{w}$ as the sole evolutionary variable in (34) concerns the stability issue of divergence control, as discussed in Sect. 2.5.2.

The choice of the functional form of $\mathfrak{F}(\mathbf{w})$ in (35) is sufficient to recover the solution of the INSE from that of (34) and (35); see Sect. 2.5.3.

2.5.2 Divergence Control

In the original PPE formulation, the only evolutionary variable is the velocity, with the pressure considered as an implicit function of it. In deriving the pressure Poisson equation (2), the divergence-free condition (1b) is applied. However, it is impossible for a numerical scheme to *exactly* enforce (1b) on computers with *finite* precision. Furthermore, it is often unnecessary for practical applications to satisfy (1b) to machine precision. Since (1b) is already assumed in deriving the governing equations of the original PPE formulation, it is difficult to analyze how an approximation of (1b) would affect the solutions of the INSE. Indeed, the momentum equation (1a) and the pressure Poisson equation (2) lead to a degenerate dynamics of velocity divergence: $\frac{\partial \nabla\cdot\mathbf{u}}{\partial t} = 0$.

In the UPPE formulation, the velocity $\mathbf{u}$ is still the sole evolutionary variable, but it is not assumed to be divergence free; the velocity divergence evolves as the solution of the heat equation with no-flux boundary conditions [33, Theorem 5.1]. This is an improvement to the original PPE formulation since the exponential decay dictated by the maximal principle of the heat equation is a much better control of the velocity divergence. However, the derivation of the UPPE formulation relies on (10). Similar to the divergence-free condition in the original PPE formulation, the properties of the Leray–Helmholtz projection cannot be satisfied *exactly* on computers with finite precision. Also, it is sometimes difficult or practically unnecessary for a discrete projection operator to fulfill (10) to machine precision. More importantly, it is not clear in the UPPE formulation how the final solution is affected by an error of approximating (10).

In comparison, the GePUP governing equations (34) are derived without (10). Of course, if the generic projection $\mathcal{P}$ in (34) is identified with the Leray–Helmholtz projection, then the assumptions in (33) clearly hold and the GePUP formulation (34) reduces to UPPE. Hence, the GePUP formulation is more general than UPPE. In the meantime, GePUP retains the advantage of UPPE since taking the divergence of (34a) and applying (34c) yield

$$\frac{\partial(\nabla\cdot\mathbf{w})}{\partial t}=\nu\Delta(\nabla\cdot\mathbf{w}). \tag{36}$$

If one imposes the Dirichlet condition $\nabla\cdot\mathbf{w}=0$ on $\partial\Omega$, the maximum principle of the heat equation dictates that $\nabla\cdot\mathbf{w}$ decays exponentially in Ω. Furthermore, *this exponential decay of* $\nabla\cdot\mathbf{w}$ *in* (36) *is a consequence of* (34), *and is independent of the assumptions in* (33) *and the choice of* $\mathfrak{F}(\mathbf{w})$ *in* (35).

Indeed, (34a) and (34c) imply that the divergence part of $\langle\mathbf{g}\rangle-\mathbf{D}\langle\mathbf{uu}\rangle$ will always be balanced by ∇q regardless of $\mathbf{u}$. Even if the assumptions (33) used in deriving (34) do not hold initially, they will be justified after a time period that is sufficiently long, because both the spatial and temporal gradients of $\nabla\cdot\mathbf{w}$ will become negligible then. However, the initial value of $\mathbf{w}$ for certain generic projections may be so large that the deviation of the solution of (34) from that of the INSE is unacceptable for the initial period of divergence decay. This issue is addressed in the next subsection.

2.5.3 Consistency with the INSE

If the generic projection $\mathcal{P}$ in (34) does not satisfy (33), the solution of (34) is different from that of the INSE. For consistency of these two solutions, we should choose $\mathcal{P}$ and $\mathfrak{F}$ so that $\mathbf{w}$ recovers $\mathbf{u}$.

The simplest set of choices is probably $\mathcal{P}=\mathscr{P}$ and (35). Then (32) and (35) yield $\mathbf{u}=\mathscr{P}^2\mathbf{w}$, which implies $\nabla\cdot\mathbf{u}=0$ by (10a). Then (32) yields $\mathbf{w}=\mathbf{u}$ and consequently $q=p$. As another advantage of $\mathcal{P}=\mathscr{P}$, the initial condition is simply $\mathbf{w}(t_0)=\mathbf{u}(t_0)$ where t_0 is the initial time.

The key point of such choices is as follows. A discretization of (34) in space involves not the generic projection in (32) but the Leray–Helmholtz projection in (35). In other words, we only need to verify that there does exist at least one choice of $\mathcal{P}$ such that $\mathbf{w}$ recovers the solution of the INSE, but we are not interested in the specific form of any such $\mathcal{P}$ (except for obtaining from (32) the value of $\mathbf{w}$ at the initial time), because (34) and (35) already form a complete evolution of $\mathbf{w}$. This is an advantage of GePUP over UPPE: the derivation of the governing equation (34) formally does not involve the Leray–Helmholtz projection $\mathscr{P}$, which only comes into the system through (35) as *one* possible choice of the auxiliary variable $\mathbf{u}$ to recover the solution of the INSE. Consequently, approximating $\mathscr{P}$ with a discrete projection only affects the accuracy of the *spatial* discretization of (35), and the error of this approximation on the dynamics of the sole evolutionary variable $\mathbf{w}$ is clear.

Note that one does not have to choose $\mathcal{P}=\mathscr{P}$ for the purpose of recovering $\mathbf{u}$ from $\mathbf{w}$. For example, let us consider an infinite sequence of generic projections $\{\mathcal{P}_n:n\in\mathbb{N}^+\}$ in which $\mathcal{P}_n\neq\mathscr{P}$ for each n, but the limit of $\{\mathcal{P}_n\}$ converges to $\mathscr{P}$:

$$\lim_{n\to+\infty}\mathcal{P}_n=\mathscr{P}. \tag{37}$$

Each generic projection $\mathcal{P}_n$ yields an initial condition of $\mathbf{w}_n(t_0)=\mathcal{P}_n\mathbf{u}_n(t_0)$, which, together with (34) and (35), recovers the solution of the INSE at the limit of $n\to+\infty$, i.e. $\forall t_e>t_0$, $\lim_{n\to+\infty}\mathbf{w}_n(t_e)=\mathbf{u}(t_e)$. In fact, even (35) is not necessary because we can simply choose $\mathbf{u}=\mathcal{P}_n\mathbf{w}$ for a sequence $\{\mathcal{P}_n:n\in\mathbb{N}^+\}$ that satisfies (37). It would be an analytic curiosity to answer whether other choices of $\{\mathcal{P}_n\}$ and $\{\mathfrak{F}_n\}$ can recover the solution of the INSE from the limit of $\{\mathbf{w}_n\}$. This question, however, is out of the scope of this work.

So far we have assumed that unique bounded solutions of the INSE do exist. Cozzi and Pego [14] showed that the solution of the INSE might not be bounded at a domain corner that is not C^3. Consequently, the velocity divergence of the INSE is not bounded either. This

is however not a contradiction to (36). Instead, the unboundedness simply implies that one cannot make the assumptions in (33) and hence (36) does not hold any more. After all, A consistent reformulation of the INSE should not change the fact that sometimes the INSE do not admit bounded solutions.

2.5.4 Summary

Distinguishing features of the GePUP formulation of the INSE with no-slip conditions are summarized as follows.

- The sole evolutionary variable in GePUP is the projected velocity $\mathbf{w}$, and the governing equations facilitate a "black-box" employment of a time integrator.
- If the initial condition of the INSE admits bounded solutions, the divergence of $\mathbf{w}$ decays exponentially.
- The Leray–Helmholtz projection $\mathscr{P}$ is not used in deriving the governing equations for the sole evolutionary variable $\mathbf{w}$. Instead, $\mathscr{P}$ only comes into the equations as one possible choice to recover from $\mathbf{w}$ the true solution of the INSE. This furnishes a simple accuracy analysis via method-of-lines.
- A discrete projection is only required to approximate $\mathscr{P}$ to certain accuracy, not to exactly satisfy its properties (10); this provides a certain degree of flexibility in the design of high-order numerical methods.

3 Spatial Discretization

3.1 Finite-Volume Discretization

The problem domain Ω is discretized into a collection of square control volumes. Denote a control volume by a multi-index $\mathbf{i} \in \mathbb{Z}^{\mathrm{D}}$, the region of cell $\mathbf{i}$ is represented by

$$\mathcal{C}_{\mathbf{i}} := \left[\mathbf{x}_O + \mathbf{i}h,\ \mathbf{x}_O + (\mathbf{i} + \mathbb{1})\, h\right], \tag{38}$$

and the region of the higher face of cell $\mathbf{i}$ in dimension d by

$$\mathcal{F}_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} := \left[\mathbf{x}_O + \left(\mathbf{i} + \mathbf{e}^d\right) h,\ \mathbf{x}_O + (\mathbf{i} + \mathbb{1})\, h\right], \tag{39}$$

where $\mathbf{x}_O \in \mathbb{R}$ is some fixed origin of the coordinates, h the uniform grid size, $\mathbb{1} \in \mathbb{Z}^{\mathrm{D}}$ the multi-index with all its components equal to one, and $\mathbf{e}^d \in \mathbb{Z}^{\mathrm{D}}$ a multi-index with 1 as its dth component and 0 otherwise. Throughout this paper, regular lowercase letters denote scalars, boldface lowercase letters vectors, boldface uppercase letters discrete operators, and calligraphic uppercase letters point sets or continuous operators.

The author strictly distinguishes three different types of values, viz. point values, cell-averaged values, and face-averaged values. As shown in Fig. 1, point values are denoted by naked symbols with the subscripts indicating their locations, e.g. $\phi_{\mathbf{i}}$ and $\phi_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}$ are the point values of ϕ at the center of the cell $\mathbf{i}$ and the center of the face $\mathbf{i} + \frac{1}{2}\mathbf{e}^d$, respectively. A symbol within a pair of angle brackets is a cell-averaged value if the subscript is an integer, or a face-averaged value if the subscript is fractional. For example, the averaged ϕ over cell $\mathbf{i}$ is denoted by

$$\langle \phi \rangle_{\mathbf{i}} := \frac{1}{h^{\mathrm{D}}} \int_{\mathcal{C}_{\mathbf{i}}} \phi\,(\mathbf{x})\ \mathrm{d}\mathbf{x}, \tag{40}$$

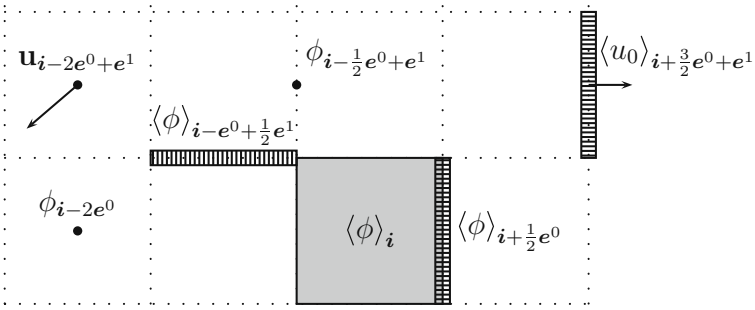

Fig. 1 The notation of point, face-averaged and cell-averaged values. u_0 is the first component of velocity vector $\mathbf{u}$. A *symbol* without the averaging operator $\langle\cdot\rangle$ denotes a point value; otherwise it denotes either a cell-averaged value if the subscript is an integer multi-index, or a face-averaged value if the subscript is a fractional multi-index. Hence $\phi_{\mathbf{i}-2\mathbf{e}^0}$, $\phi_{\mathbf{i}-\frac{1}{2}\mathbf{e}^0+\mathbf{e}^1}$, and $\mathbf{u}_{\mathbf{i}-2\mathbf{e}^0+\mathbf{e}^1}$ are point values; $\langle\phi\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}$, $\langle\phi\rangle_{\mathbf{i}-\mathbf{e}^0+\frac{1}{2}\mathbf{e}^1}$, and $\langle u_0\rangle_{\mathbf{i}+\frac{3}{2}\mathbf{e}^0+\mathbf{e}^1}$ are face-averaged values; $\langle\phi\rangle_{\mathbf{i}}$ is a cell-averaged value. *Horizontal* and *vertical hatches* represent the averaging processes over a *vertical cell face* and a *horizontal cell face*, respectively. *Light gray* area represents averaging over a cell

and the averaged ϕ over the face $\mathbf{i}+\frac{1}{2}\mathbf{e}^d$ by

$$\langle\phi\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} := \frac{1}{h^{\mathrm{D}-1}}\int_{\mathcal{F}_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}} \phi\,(\mathbf{x})\;\mathrm{d}\mathbf{x}. \tag{41}$$

Point values can be converted to face-averaged and cell-averaged values and vice versa by Taylor expansions. For example, expanding the integrands in (40) and (41) yields

$$\langle\phi\rangle_{\mathbf{i}} = \phi_{\mathbf{i}} + \frac{h^2}{24}\sum_{d=1}^{\mathrm{D}} \left.\frac{\partial^2\phi(\mathbf{x})}{\partial x_d^2}\right|_{\mathbf{i}} + O(h^4); \tag{42a}$$

$$\langle\phi\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} = \phi_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} + \frac{h^2}{24}\sum_{d'\neq d} \left.\frac{\partial^2\phi(\mathbf{x})}{\partial x_{d'}^2}\right|_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} + O(h^4). \tag{42b}$$

Relations between cell averages and face averages can be derived via the first fundamental theorem of calculus; two formulas [60, Appendix A] used throughout this work are

$$\langle\phi\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} = \frac{1}{12}\Big(-\langle\phi\rangle_{\mathbf{i}+2\mathbf{e}^d} + 7\,\langle\phi\rangle_{\mathbf{i}+\mathbf{e}^d} + 7\langle\phi\rangle_{\mathbf{i}} - \langle\phi\rangle_{\mathbf{i}-\mathbf{e}^d}\Big) + O(h^4); \tag{43a}$$

$$\left\langle\frac{\partial\phi}{\partial x_d}\right\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} = \frac{1}{12h}\Big(-\langle\phi\rangle_{\mathbf{i}+2\mathbf{e}^d} + 15\,\langle\phi\rangle_{\mathbf{i}+\mathbf{e}^d} - 15\langle\phi\rangle_{\mathbf{i}} + \langle\phi\rangle_{\mathbf{i}-\mathbf{e}^d}\Big) + O(h^4). \tag{43b}$$

Since the leading term of the difference between cell-centered and cell-averaged values is $O(h^2)$, there is no need to distinguish these two types of centering in second-order methods. In contrast, it is necessary to to treat them differently in fourth-order methods. Furthermore, finite-volume operators acting on averaged values are different from their finite-difference counterparts acting on point values. The difference between the three types of values account for much of the complexity of fourth-order finite-volume methods that is absent in second-order methods.

3.2 Discrete Operators that Act on Cell Averages

The discrete gradient, the discrete divergence, and the discrete Laplacian are

$$\mathbf{G}_d \langle\phi\rangle_{\mathbf{i}} := \frac{1}{12h}\Big(-\langle\phi\rangle_{\mathbf{i}+2\mathbf{e}^d} + 8\langle\phi\rangle_{\mathbf{i}+\mathbf{e}^d} - 8\langle\phi\rangle_{\mathbf{i}-\mathbf{e}^d} + \langle\phi\rangle_{\mathbf{i}-2\mathbf{e}^d}\Big), \tag{44}$$

$$\mathbf{D}\langle\mathbf{u}\rangle_{\mathbf{i}} := \frac{1}{12h}\sum_d\Big(-\langle u_d\rangle_{\mathbf{i}+2\mathbf{e}^d} + 8\langle u_d\rangle_{\mathbf{i}+\mathbf{e}^d} - 8\langle u_d\rangle_{\mathbf{i}-\mathbf{e}^d} + \langle u_d\rangle_{\mathbf{i}-2\mathbf{e}^d}\Big), \tag{45}$$

$$\mathbf{L}\langle\phi\rangle_{\mathbf{i}} := \frac{1}{12h^2}\sum_d\Big(-\langle\phi\rangle_{\mathbf{i}+2\mathbf{e}^d} + 16\langle\phi\rangle_{\mathbf{i}+\mathbf{e}^d} - 30\langle\phi\rangle_{\mathbf{i}} + 16\langle\phi\rangle_{\mathbf{i}-\mathbf{e}^d} - \langle\phi\rangle_{\mathbf{i}-2\mathbf{e}^d}\Big). \tag{46}$$

The discrete divergence also acts on tensor averages

$$\mathbf{D}\langle\mathbf{uu}\rangle_{\mathbf{i}} := \frac{1}{h}\sum_d\left(\mathbf{F}\langle u_d, \mathbf{u}\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} - \mathbf{F}\langle u_d, \mathbf{u}\rangle_{\mathbf{i}-\frac{1}{2}\mathbf{e}^d}\right), \tag{47}$$

where the average of the product of two scalars over a cell face is

$$\mathbf{F}\langle\varphi,\psi\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} := \langle\varphi\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}\langle\psi\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} + \frac{h^2}{12}\sum_{d'\neq d}\left(\mathbf{G}^{\perp}_{d'}\varphi\right)_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}\left(\mathbf{G}^{\perp}_{d'}\psi\right)_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}, \tag{48}$$

and $\mathbf{G}^{\perp}_{d'}$ is the discrete gradient in the transverse directions

$$\left(\mathbf{G}^{\perp}_{d'}\varphi\right)_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} := \frac{1}{2h}\left(\langle\varphi\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d+\mathbf{e}^{d'}} - \langle\varphi\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d-\mathbf{e}^{d'}}\right). \tag{49}$$

$\mathbf{G}\langle\phi\rangle$, $\mathbf{D}\langle\mathbf{u}\rangle$, $\mathbf{L}\langle\mathbf{u}\rangle$, and $\mathbf{D}\langle\mathbf{uu}\rangle$ approximate the cell average of a gradient, velocity divergence, Laplacian, and convection, respectively. These discrete operators are formally fourth-order accurate in approximating their continuous counterparts; see [57, Proposition 1][2] for a precise statement.

By using the same arguments in [60, Appendix B], one can prove

$$\frac{1}{h^{\mathrm{D}}}\int_{\mathcal{C}_{\mathbf{i}}}\varphi\psi = \langle\varphi\rangle_{\mathbf{i}}\langle\psi\rangle_{\mathbf{i}} + \frac{1}{48}\sum_{d=1}^{\mathrm{D}}\left(\langle\varphi\rangle_{\mathbf{i}+\mathbf{e}^d} - \langle\varphi\rangle_{\mathbf{i}-\mathbf{e}^d}\right)\left(\langle\psi\rangle_{\mathbf{i}+\mathbf{e}^d} - \langle\psi\rangle_{\mathbf{i}-\mathbf{e}^d}\right) + O(h^4). \tag{50}$$

Define a discrete inner product of two vector functions by their cell averages over the domain Ω,

$$\langle\mathbf{u},\mathbf{v}\rangle_V := h^{\mathrm{D}}\sum_{\mathcal{C}_{\mathbf{i}}\subset\Omega}\langle\mathbf{u}\rangle_{\mathbf{i}}\cdot\langle\mathbf{v}\rangle_{\mathbf{i}}. \tag{51}$$

It is not difficult to show that $\langle\mathbf{u},\mathbf{v}\rangle_V = O(h^4)$ is a sufficient condition for a fourth-order fulfillment of the L_2-orthogonality of $\mathbf{u}$ and $\mathbf{v}$, i.e.

$$\langle\mathbf{u},\mathbf{v}\rangle_V = O(h^4) \;\Rightarrow\; \int_{\Omega}\mathbf{u}\cdot\mathbf{v} = O(h^4). \tag{52}$$

[2] Equation (14e) in [57] contains an error; it should be $\mathbf{F}\langle\varphi,\psi\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} = \frac{1}{h^{\mathrm{D}-1}}\int_{\mathcal{F}_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}}\varphi\psi + O(h^4)$. The author regrets the oversight of dropping the constant $\frac{1}{h^{\mathrm{D}-1}}$ in the RHS.

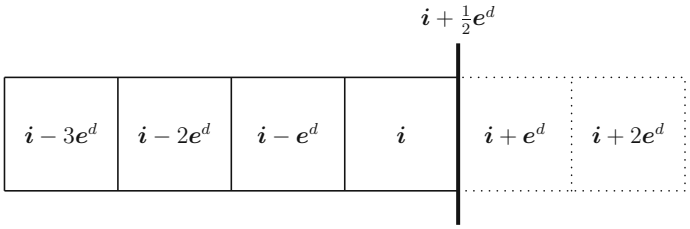

Fig. 2 An example ($d = 1$) of ghost filling for the domain boundary. The *thick line* represents a physical boundary, the *thin solid lines* interior cells, and the *dotted lines* ghost cells

3.3 Evaluating Discrete Operators Via Ghost Cells

Following [57], two layers of ghost cells are used to enforce boundary conditions. For non-periodic boundaries, the values of ghost cells are obtained by extrapolating those of the interior cells, with the boundary conditions incorporated in the extrapolation formulas. Referring to Fig. 2, different boundary conditions entails different cell-averaged values for the cells $\mathbf{i}+\mathbf{e}^d$ and $\mathbf{i}+2\mathbf{e}^d$. For example, homogeneous Dirichlet boundary conditions for a scalar ψ are fulfilled by filling the ghost cells with the following fifth-order formulas:

$$\begin{aligned}\langle\psi\rangle_{\mathbf{i}+\mathbf{e}^d} &= \tfrac{1}{12}\left(-77\langle\psi\rangle_{\mathbf{i}} + 43\langle\psi\rangle_{\mathbf{i}-\mathbf{e}^d} - 17\langle\psi\rangle_{\mathbf{i}-2\mathbf{e}^d} + 3\langle\psi\rangle_{\mathbf{i}-3\mathbf{e}^d}\right) + O(h^5),\\ \langle\psi\rangle_{\mathbf{i}+2\mathbf{e}^d} &= \tfrac{1}{12}\left(-505\langle\psi\rangle_{\mathbf{i}} + 335\langle\psi\rangle_{\mathbf{i}-\mathbf{e}^d} - 145\langle\psi\rangle_{\mathbf{i}-2\mathbf{e}^d} + 27\langle\psi\rangle_{\mathbf{i}-3\mathbf{e}^d}\right) + O(h^5).\end{aligned} \tag{53}$$

Neumann boundary conditions are fulfilled by

$$\begin{aligned}\langle\psi\rangle_{\mathbf{i}+\mathbf{e}^d} &= \tfrac{1}{10}\left(5\langle\psi\rangle_{\mathbf{i}} + 9\langle\psi\rangle_{\mathbf{i}-\mathbf{e}^d} - 5\langle\psi\rangle_{\mathbf{i}-2\mathbf{e}^d} + \langle\psi\rangle_{\mathbf{i}-3\mathbf{e}^d}\right)\\ &\quad +\tfrac{6}{5}h\left\langle\tfrac{\partial\psi}{\partial n}\right\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} + O(h^5),\\ \langle\psi\rangle_{\mathbf{i}+2\mathbf{e}^d} &= \tfrac{1}{10}\left(-75\langle\psi\rangle_{\mathbf{i}} + 145\langle\psi\rangle_{\mathbf{i}-\mathbf{e}^d} - 75\langle\psi\rangle_{\mathbf{i}-2\mathbf{e}^d} + 15\langle\psi\rangle_{\mathbf{i}-3\mathbf{e}^d}\right)\\ &\quad +6h\left\langle\tfrac{\partial\psi}{\partial n}\right\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} + O(h^5),\end{aligned} \tag{54}$$

where $\left\langle\frac{\partial\psi}{\partial n}\right\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}$ is the Neumann condition for ψ. Note that $\frac{\partial\psi}{\partial n}$ instead of $\frac{\partial\psi}{\partial x_d}$ is used here so that the formulas remain the same for boundaries at the lower sides.

When no boundary conditions are known, a scalar ψ can be smoothly extended to fill a ghost cell abutting the boundary,

$$\langle\psi\rangle_{\mathbf{i}+\mathbf{e}^d} = 5\langle\psi\rangle_{\mathbf{i}} - 10\langle\psi\rangle_{\mathbf{i}-\mathbf{e}^d} + 10\langle\psi\rangle_{\mathbf{i}-2\mathbf{e}^d} - 5\langle\psi\rangle_{\mathbf{i}-3\mathbf{e}^d} + \langle\psi\rangle_{\mathbf{i}-4\mathbf{e}^d} + O(h^5); \tag{55}$$

its faced-averaged value at the boundary can also be obtained similarly as

$$\begin{aligned}\langle\psi\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} = \tfrac{1}{60}&\left(137\langle\psi\rangle_{\mathbf{i}} - 163\langle\psi\rangle_{\mathbf{i}-\mathbf{e}^d} + 137\langle\psi\rangle_{\mathbf{i}-2\mathbf{e}^d}\right.\\ &\left.-63\langle\psi\rangle_{\mathbf{i}-3\mathbf{e}^d} + 12\langle\psi\rangle_{\mathbf{i}-4\mathbf{e}^d}\right) + O(h^5).\end{aligned} \tag{56}$$

In comparison, face-averaged derivatives can be calculated from known boundary conditions and interior cell averages:

$$\left\langle \frac{\partial \psi}{\partial n} \right\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} = \frac{1}{72h}\Big(-415\langle\psi\rangle_{\mathbf{i}} + 161\langle\psi\rangle_{\mathbf{i}-\mathbf{e}^d} - 55\langle\psi\rangle_{\mathbf{i}-2\mathbf{e}^d} + 9\langle\psi\rangle_{\mathbf{i}-3\mathbf{e}^d} + 300\langle\psi\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}\Big) + O(h^4), \tag{57}$$

$$\left\langle \frac{\partial^2 \psi}{\partial^2 n} \right\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} = \frac{1}{48h^2}\Big(-755\langle\psi\rangle_{\mathbf{i}} + 493\langle\psi\rangle_{\mathbf{i}-\mathbf{e}^d} - 191\langle\psi\rangle_{\mathbf{i}-2\mathbf{e}^d} + 33\langle\psi\rangle_{\mathbf{i}-3\mathbf{e}^d} + 420\langle\psi\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}\Big) + O(h^3). \tag{58}$$

With the above ghost-filling formulas, the evaluation of a discrete operator mainly consists of two steps: fill the ghost cells of the input cell-averages and then apply stencils such as those of $\mathbf{G}_d$, $\mathbf{D}$, and $\mathbf{L}$.

The steps of evaluating the discrete convection $\mathbf{D}\langle\mathbf{uu}\rangle$ are summarized as follows:

(Cnv-1) fill the ghost cells[3] of cell-averaged velocity $\langle\mathbf{u}\rangle_{\mathbf{i}}$ using (53),
(Cnv-2) convert $\langle\mathbf{u}\rangle_{\mathbf{i}}$ to face-averaged *normal*[4] velocity $\langle u_d\rangle_{\mathbf{i}\pm\frac{1}{2}\mathbf{e}^d}$ for each dimension d using (43a),
(Cnv-3) convert $\langle\mathbf{u}\rangle_{\mathbf{i}}$ to face-averaged velocity $\langle\mathbf{u}\rangle_{\mathbf{i}\pm\frac{1}{2}\mathbf{e}^d}$ for *each* component of the velocity and each dimension d using (43a),
(Cnv-4) calculate the discrete velocity product $\mathbf{F}(u_d, \mathbf{u})$ using (48),
(Cnv-5) compute $\mathbf{D}\langle\mathbf{uu}\rangle$ using (47).

All formulas presented in this subsection can be generated via fitting one-dimensional polynomials of the fifth order; a MATLAB implementation is included in the Appendix. Due to the fifth-order accuracy, the discrete operators with first derivatives have fourth-order truncation errors. For the discrete Laplacian $\mathbf{L}$, numerical experiments show that six-order ghost-filling formulas result in excessive numerical oscillation and less accurate solutions. Hence the aforementioned fifth-order formulas are also used for evaluating $\mathbf{L}$, although the truncation error of $\mathbf{L}$ is only of the third order near the domain boundary.

3.4 Pressure Extraction From the Velocity (PEV)

In this subsection we discuss the procedures for extracting cell averages of the pressure q from those of the velocity $\mathbf{u}$.

It follows from (34c) and $\int_\Omega \nabla\cdot(\mathbf{u}\cdot\nabla\mathbf{u}) = \oint_{\partial\Omega}\mathbf{n}\cdot(\mathbf{u}\cdot\nabla\mathbf{u}) = 0$ that

$$\oint_{\partial\Omega}\mathbf{n}\cdot\nabla q = \oint_{\partial\Omega}\mathbf{n}\cdot\mathbf{g}, \tag{59}$$

which agrees with (34d) since $\oint_{\partial\Omega}\mathbf{n}\cdot(\nu\Delta\mathbf{u} - \nu\nabla\nabla\cdot\mathbf{u}) = \int_\Omega \nabla\cdot\mathcal{B}\mathbf{u} = 0$. Nonetheless, (34d) and (59) are different in that the former is a local relation while the latter a global constraint.

The discrete operators in Sect. 3.2 can be employed to discretize (34c) into a linear system,

$$\mathbf{L}\langle q\rangle = \mathbf{D}\big(\langle\mathbf{g}\rangle - \mathbf{D}\langle\mathbf{uu}\rangle\big), \tag{60}$$

[3] Due to the transverse gradient (49), the ghost cells diagonal to domain corners also need to be filled. For this purpose, the off-diagonal ghost cells are filled first, their values are then plugged into (53) to calculate those of the diagonal ghost cells.

[4] In other words, each cell face only has the single velocity component normal to it. The face-averaged velocity here is the same as that on staggered grids. In comparison, each face in (Cnv-3) has all the components of velocity averaged over it.

where the unknown is the vector of cell averages of q. (60) is complemented by an inhomogeneous Neumann condition that results from averaging (34d) on cell faces $\mathcal{F}_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} \subset \partial\Omega$,

$$\left\langle \frac{\partial q}{\partial n} \right\rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} = \langle g_n \rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} + \nu \langle \Delta u_n \rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} - \nu \left\langle \frac{\partial \nabla \cdot \mathbf{u}}{\partial n} \right\rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n}, \tag{61}$$

where a subscript "$_n$" denotes the normal component of a vector.

The inhomogeneous Neumann condition (61) can be converted to a homogeneous one by moving its contribution to the RHS of (60). For the example in Fig. 2, this is done by adding[5] $-\frac{11}{10h}\left\langle \frac{\partial q}{\partial n} \right\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}$ to $\langle \nabla \cdot (\mathbf{g} - \mathbf{u} \cdot \nabla \mathbf{u}) \rangle_{\mathbf{i}}$ and adding $\frac{1}{10h}\left\langle \frac{\partial q}{\partial n} \right\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}$ to $\langle \nabla \cdot (\mathbf{g} - \mathbf{u} \cdot \nabla \mathbf{u}) \rangle_{\mathbf{i}-\mathbf{e}^d}$. Clearly, we need to calculate $\left\langle \frac{\partial q}{\partial n} \right\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}$ for all the boundary faces.

The term $\langle g_n \rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n}$ in (61) is known a priori. For the nonhomogeneous value of the Neumann condition, we still need to compute $\left\langle \frac{\partial \nabla \cdot \mathbf{u}}{\partial n} \right\rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n}$ and $\langle \Delta u_n \rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n}$ on all boundary faces $\mathcal{F}_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n}$. The former is obtained by first calculating $\langle \nabla \cdot \mathbf{u} \rangle_{\mathbf{i}}$ for cells near the boundary and then applying (57) with $\langle \nabla \cdot \mathbf{u} \rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} = 0$ on $\partial\Omega$. As for the latter, we first note that, on a *flat* no-penetration boundary with $\nabla \cdot \mathbf{u} = 0$ and $u_\tau(\tau, t) = u(t)$ (uniform tangential velocity), the only nonzero component of the velocity gradient tensor is $\frac{\partial u_\tau}{\partial n}$, i.e.

$$\frac{\partial u_n}{\partial \tau} = 0, \quad \frac{\partial u_\tau}{\partial \tau} = 0, \quad \frac{\partial u_n}{\partial n} = 0, \quad \text{on } \partial\Omega. \tag{62}$$

Then $\frac{\partial u_n}{\partial \tau} = 0$ leads to the simplification

$$\langle \Delta u_n \rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} = \left\langle \frac{\partial^2 u_n}{\partial n^2} \right\rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n}$$

so that one could use (58) with $\langle u_n \rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} = 0$ to calculate the above RHS.

Solving (60) with an iterative solver essentially conerns computing its residual, i.e. the difference between its LHS and RHS, from the given cell averages $\langle \mathbf{u} \rangle$ and $\langle q \rangle$. The steps of residual computation for (60) are summarized as follows,

(PEV-1) compute the discrete convection in the RHS of (60) with homogeneous Dirichlet conditions $\langle \mathbf{u} \rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} = \mathbf{0}$ by the steps (Cnv-1) to (Cnv-5) as detailed in Sect. 3.3,

(PEV-2) fill the ghost cells of the normal component of cell-averaged convection with homogeneous Dirichlet conditions by (53),

(PEV-3) convert the discrete convection to normal face averages using (43a), evaluate the divergence of the face-averaged normal convection by

$$\mathbf{DD} \langle \mathbf{uu} \rangle_{\mathbf{i}} = \sum_d \left(\langle \mathbf{D} \langle \mathbf{uu} \rangle \rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} - \langle \mathbf{D} \langle \mathbf{uu} \rangle \rangle_{\mathbf{i}-\frac{1}{2}\mathbf{e}^d} \right), \tag{63}$$

and add to it $\mathbf{D} \langle \mathbf{g} \rangle$ to obtain the RHS of (60),

(PEV-4) calculate the Neumann value $\left\langle \frac{\partial q}{\partial n} \right\rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n}$ for all cell faces $\mathcal{F}_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} \subset \partial\Omega$,

(PEV-5) enforce the global constraint (59) to machine precision by subtracting a small constant[6] from the Neumann value of each boundary face,

[5] The coefficients $-\frac{11}{10h}$ and $-\frac{11}{10h}$ can be easily derived from (46) and (54).

[6] This constant is the average of these Neumann boundary values. For all numerical experiments in Sect. 6, no relative changes of $\left\langle \frac{\partial q}{\partial n} \right\rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n}$ are greater than 0.005.

(PEV-6) add to the RHS computed in (PEV-3) the contribution of the nonhomogenous Neumann condition (61),
(PEV-7) fill the ghost cells of $\langle q \rangle$ using (54) with *homogeneous* Neumann condition $\left\langle \frac{\partial q}{\partial n} \right\rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} = 0$,
(PEV-8) evaluate the discrete Laplacian in the LHS of (60) and subtract from it the RHS in (PEV-6) to obtain the residual.

The enforcement of the global constraint (59) in (PEV-5) satisfies to machine precision the solvability condition of (34c) and (34d), enhances numerical stability, and improve the performance of the geometric multigrid method; see Sect. 5.1 for more details on this linear solver. Note that it is an advantage of the finite-volume formulation that enable us to enforce (59) to machine precision: the sum of all cell averages of a variable scaled by h^{D} is formally the integral of that variable over the whole domain.

4 A Fourth-Order Discrete Projection

A discrete projection can be formed using the discrete operators in Sect. 3.2:

$$\mathbf{P} := \mathbf{I} - \mathbf{G}\mathbf{L}^{-1}\mathbf{D}. \tag{64}$$

On periodic domains, $\mathbf{P}$ is a fourth-order approximate to the Leray–Helmholtz projection [57, Theorem 5]. $\mathbf{P}$ is *approximate* in the sense that $\mathbf{P}^2 \neq \mathbf{P}$, which follows from $\mathbf{L} \neq \mathbf{D}\mathbf{G}$. The fact of $\mathbf{P}^2 \neq \mathbf{P}$ might be disturbing to the reader; after all, $\mathbf{L}$ equals $\mathbf{D}\mathbf{G}$ on periodic staggered grids and the corresponding discrete projection is indeed idempotent. However, due to the non-periodicity of the domain and the requirement of fourth-order accuracy, it is very difficult, even on staggered grids, to satisfy the idempotence condition on no-slip domains. In other words, even if the pressure and the velocity are staggered, the discrete projection is still approximate in the sense that $\mathbf{P}^2 \neq \mathbf{P}$. Also, $\mathbf{P}$ does not satisfy other identities in (3). Fortunately, $\mathbf{P}$ can be considered as a generic projection on a finite-dimensional vector space and the GePUP formulation provides the appropriate analytic framework for the use of it. Furthermore, as far as the fourth-order accuracy is concerned, the inexactness of the discrete projection is not problematic because the divergence of the projected velocity in the GePUP formulation (34) decays exponentially, as discussed in Sect. 2.5.

4.1 Implementation for No-Penetration Domains

For a Leray–Helmholtz projection on no-penetration domains, we have

$$\mathscr{P}\mathbf{u} = \mathbf{u} - \nabla\phi, \tag{65a}$$
$$\Delta\phi = \nabla \cdot \mathbf{u}, \tag{65b}$$
$$\mathbf{n} \cdot \nabla\phi = \mathbf{n} \cdot \mathbf{u}. \tag{65c}$$

The integration of (65b) over a control volume $\mathcal{C}_\mathbf{i}$ yields

$$\sum_d \left(\langle u_d \rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} - \langle u_d \rangle_{\mathbf{i}-\frac{1}{2}\mathbf{e}^d} \right) = \sum_d \left(\left\langle \frac{\partial \phi}{\partial x_d} \right\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d} - \left\langle \frac{\partial \phi}{\partial x_d} \right\rangle_{\mathbf{i}-\frac{1}{2}\mathbf{e}^d} \right).$$

For the example shown in Fig. 2, (65c) further simplifies the above equation to

$$\langle u_2\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^2} - \langle u_2\rangle_{\mathbf{i}-\frac{1}{2}\mathbf{e}^2} - \langle u_1\rangle_{\mathbf{i}-\frac{1}{2}\mathbf{e}^1} = \left\langle \frac{\partial \phi}{\partial x_2} \right\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^2} - \left\langle \frac{\partial \phi}{\partial x_2} \right\rangle_{\mathbf{i}-\frac{1}{2}\mathbf{e}^2} - \left\langle \frac{\partial \phi}{\partial x_1} \right\rangle_{\mathbf{i}-\frac{1}{2}\mathbf{e}^1}.$$

Thanks to the special form of (65b) and (65c), $\langle u_1\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^1}$ will always be canceled by $\left\langle \frac{\partial \phi}{\partial x_1} \right\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^1}$, regardless of its actual value. Therefore, a fourth-order approximation to the BVP of (65b) and (65c) is

$$\mathbf{L}_H \langle \phi \rangle = \mathbf{D}_H \langle \mathbf{u} \rangle, \tag{66}$$

where $\mathbf{L}_H, \mathbf{D}_H$ are the same as $\mathbf{L}, \mathbf{D}$ defined in (46) and (45), except that the fluxes at the wall boundaries are set to zero. In a projection method for moving quadrilateral grids, Trebotich and Colella [51] also converted an inhomogeneous Neumann BVP to a homogeneous one.

In this work, we calculate $\mathbf{D}_H \langle \mathbf{u} \rangle$ as follows,

(DvH-1) smoothly extend $\langle u_n \rangle$ to the ghost cells abutting the domain by (55),
(DvH-2) convert $\langle u_n \rangle$ to face averages using (43a),
(DvH-3) zero out the face averages on the wall boundaries,
(DvH-4) sum up the face-averaged normal velocities for each interior cell.

$\mathbf{L}_H \langle \phi \rangle$ can also be calculated by first converting cell averages of $\nabla \phi$ to face averages and then apply the above steps, but this is not amenable to a multigrid solver. To facilitate the smoothing operations, $\mathbf{L}_H \langle \phi \rangle$ is evaluated by (46), with ghost cells that immediately abut the wall filled by (55) and those away from the wall by

$$\langle \phi \rangle_{\mathbf{i}+2\mathbf{e}^d} = 60\langle \phi \rangle_{\mathbf{i}} - 149 \langle \phi \rangle_{\mathbf{i}-\mathbf{e}^d} + 150 \langle \phi \rangle_{\mathbf{i}-2\mathbf{e}^d} - 75 \langle \phi \rangle_{\mathbf{i}-3\mathbf{e}^d} + 15 \langle \phi \rangle_{\mathbf{i}-4\mathbf{e}^d} + O(h^5) \tag{67}$$

so that the resulting flux $\left\langle \frac{\partial \phi}{\partial x_d} \right\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}$ given by (43b) is zero.

The aforementioned algorithms for calculating $\mathbf{D}_H \langle \mathbf{u} \rangle$ and $\mathbf{L}_H \langle \phi \rangle$ have three notable benefits.

- Each face-averaged normal velocity inside the domain appears twice in (DvH-4), one to be added and the other to be subtracted. Consequently, (DvH-3) and (DvH-4) dictate that $\Sigma_{\mathcal{C}_\mathbf{i} \subset \Omega} \mathbf{D}_H \langle \mathbf{u} \rangle_\mathbf{i} = 0$ holds, implying an *exact* enforcement of the no-flux condition $\int_{\partial\Omega} \mathbf{n} \cdot \mathbf{u} = 0$.
- The stencil of $\mathbf{L}_H$ is the same as that of $\mathbf{L}$; this encourages software reuse of multigrid solvers.
- The solvability condition of (66) is satisfied to machine precision by (67).

In case $\mathbf{G} \langle \phi \rangle$ is needed, the ghost cells of $\langle \phi \rangle$ are filled by the formulas in (54), where the Neumann boundary value $\left\langle \frac{\partial \phi}{\partial x_d} \right\rangle_{\mathbf{i}+\frac{1}{2}\mathbf{e}^d}$ is obtained by extrapolating cell-averaged normal velocity $\langle u_n \rangle$'s to the wall boundary using (56).

In summary, the discrete projection operator employed in this work is

$$\mathbf{P} = \mathbf{I} - \mathbf{G}\mathbf{L}_H^{-1}\mathbf{D}_H, \tag{68}$$

which is a fourth-order approximation of $\mathscr{P}$ because of the stencils (44), (45), and (46), the implementation of $\mathbf{L}_H$ and $\mathbf{D}_H$ as discussed above, and the definition of the Leray–Helmholtz projection $\mathscr{P}$ in Sect. 2.1. This order-of-accuracy is confirmed by results of numerical tests in Sect. 6.1.

4.2 The Spectral Radius of P Is One

A necessary condition for numerical stability is that the spectral radius of the discrete projection be no greater than one. Due to the locally altered forms of the discrete operators at the domain boundary, a theoretical analysis is difficult. In this subsection, the spectral radius of the corresponding matrix is computed numerically on a square no-penetration domain.

Let N denote the number of cells along each dimension. Let U denote the vector containing all components of $\langle \mathbf{u} \rangle$ on the N^{D} interior cells and Φ that of all $\langle \phi \rangle$'s. Denote $N_A := N^{\mathrm{D}} + N_B + 1$ and $N_B := 2\mathrm{D}N^{\mathrm{D}-1}$. Define an augmented Laplacian $\mathbf{L}_A \in \mathbb{R}^{N_A \times N_A}$, an augmented divergence $\mathbf{D}_A \in \mathbb{R}^{N_A \times \mathrm{D}N^{\mathrm{D}}}$, and an augmented vector $\Phi_A \in \mathbb{R}^{N_A}$ as

$$\mathbf{L}_A := \begin{bmatrix} \mathbf{L}_H & \mathbf{r} & \mathbf{0} \\ \mathbb{1}^T & 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \quad \mathbf{D}_A := \begin{bmatrix} \mathbf{D}_H \\ \mathbf{0} \\ \mathbf{T} \end{bmatrix}, \quad \Phi_A := \begin{bmatrix} \Phi \\ 0 \\ U_n \end{bmatrix}, \tag{69}$$

where $\mathbf{r} \in \mathbb{R}^{N^{\mathrm{D}}}$ is a random vector to make $\mathbf{L}_A$ non-singular, $\mathbb{1} \in \mathbb{R}^{N^{\mathrm{D}}}$ a vector whose components are all 1's, $\mathbf{I} \in \mathbb{R}^{N_B \times N_B}$ an identity matrix, and $\mathbf{T} \in \mathbb{R}^{N_B \times \mathrm{D}N^{\mathrm{D}}}$ the extrapolation matrix from (56) that maps the vector U to U_n, its normal components at the domain boundary. It follows from (69) and the construction of $\mathbf{P}$ in the previous subsection that the following linear system is equivalent to (66):

$$\mathbf{L}_A \Phi_A = \mathbf{D}_A U. \tag{70}$$

Define an augmented gradient matrix $\mathbf{G}_A \in \mathbb{R}^{\mathrm{D}N^{\mathrm{D}} \times N_A}$

$$\mathbf{G}_A := \begin{bmatrix} \mathbf{G}_H & 0 & \mathbf{G}_B \end{bmatrix}, \tag{71}$$

with $\mathbf{G}_B$ affording the inhomogeneous part of Neumann boundary condition for pressure gradient evaluation. The augmented projection matrix $\mathbf{P}_A \in \mathbb{R}^{\mathrm{D}N^{\mathrm{D}} \times \mathrm{D}N^{\mathrm{D}}}$,

$$\mathbf{P}_A := \mathbf{I}_A - \mathbf{G}_A \mathbf{L}_A^{-1} \mathbf{D}_A, \tag{72}$$

is then equivalent to $\mathbf{P}$ in (68). Note that the boundary conditions discussed in the previous subsection have already been incorporated inside $\mathbf{P}_A$.

The spectral radius $\rho(\mathbf{P}_A)$ is numerically computed on four two-dimensional uniform grids 16×16, 32×32, 64×64, and 128×128. As the main result, $\rho(\mathbf{P}_A) = 1.0$ holds within machine precision for each grid. In Sect. 6.1, this is confirmed by repeatedly applying $\mathbf{P}$ to a vector field and observing the convergence of the projected vector.

4.3 Pressure Extraction From Eulerian Accelerations (PEEA)

As another utility of the discrete projection, the pressure can be extracted from Eulerian accelerations (26) by rewriting the momentum equation in the INSE as

$$\mathbf{a}^* = \mathbf{a} + \nabla p \tag{73}$$

and applying $\mathbf{P}$ with the no-penetration condition $\mathbf{n} \cdot \mathbf{a} = 0$ on the domain boundary.

It is informative to compare pressure extraction from the velocity (PEV) as discussed in Sect. 3.4 to pressure extraction from Eulerian accelerations (PEEA). First, the Neumann BVP for pressure in PEV is inhomogeneous while that in PEEA is essentially homogeneous. Consequently, PEV requires nontrivial extrapolation from interior cell averages of the velocity

to its derivatives on the domain boundary for the value of $\frac{\partial q}{\partial n}$. In comparison, the boundary condition for the pressure p are not needed[7] in PEEA, as justified in the derivation of (66).

The second difference between PEV and PEEA concerns accuracy. For PEV, the extrapolated ∇q at the domain boundary is only second-order accurate because of the lop-sided extrapolation (58) and the fourth-order accuracy of the cell averages in the domain interior. Consequently, it can be shown via a Green-function analysis [32, Sect. 2.11] that the scalar q is only second-order accurate in the worst case.[8] For PEEA, the influence of the diffusion at the boundary is minimized by the homogeneous Neumann BVP (66). As a benefit, the extracted pressure p could be more accurate than q; this is confirmed in Sect. 6.

Although q and p are essentially the same in the limit of $\mathbf{w} \to \mathbf{u}$, they are distinguished in this work: q is regarded as an auxiliary variable for *evolving* the velocity while p an *instantaneous* field that results from the velocity. As discussed in Sects. 1 and 2, the strategic definition of ∇q in the GePUP formulation (34) leads to the heat equation (36) on the divergence of $\mathbf{w}$, while ∇p in the original PPE formulation yields a degenerate evolution of velocity divergence. To sum up, the distinction of p and q aims to combine the stability advantage of q with the accuracy advantage of p.

5 Temporal Integration

Integrate (34a), (34c), and (35) over the cells, approximate the integrals with the discrete operators in Sects. 3 and 4, and we have the following ODE system,

$$\frac{\mathrm{d}\,\langle \mathbf{w}\rangle}{\mathrm{d}t} = \langle \mathbf{g}\rangle - \mathbf{D}\,\langle \mathbf{u}\mathbf{u}\rangle - \mathbf{G}\,\langle q\rangle + \nu \mathbf{L}\,\langle \mathbf{w}\rangle\,, \tag{74a}$$

$$\mathbf{L}\,\langle q\rangle = \mathbf{D}\big(\langle \mathbf{g}\rangle - \mathbf{D}\,\langle \mathbf{u}\mathbf{u}\rangle\big), \tag{74b}$$

$$\langle \mathbf{u}\rangle = \mathbf{P}\,\langle \mathbf{w}\rangle\,, \tag{74c}$$

where the discrete unknown is the cell-averaged projected velocity $\langle \mathbf{w}\rangle$ and other cell-averaged quantities are considered as auxiliary functions of $\langle \mathbf{w}\rangle$. To apply a time-marching scheme to (74), we supplement it with the initial condition

$$\langle \mathbf{w}\rangle\,(t_0) = \langle \mathbf{u}\rangle\,(t_0) \tag{75}$$

and the boundary conditions

$$\langle \mathbf{w}\rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} = \mathbf{0}, \quad \langle \mathbf{u}\rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} = \mathbf{0}, \tag{76a}$$

$$\left\langle \frac{\partial q}{\partial n} \right\rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} = \langle g_n\rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} + \langle \nu \Delta u_n\rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} - \nu \left\langle \frac{\partial \nabla\cdot\mathbf{u}}{\partial n} \right\rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n}, \tag{76b}$$

$$\langle \nabla\cdot\mathbf{u}\rangle_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} = 0, \tag{76c}$$

where a subscipt "$_n$" denotes the normal component of a vector, and (76a) and (76b) follow from averaging (34b) and (34d) on a boundary face $\mathcal{F}_{\mathbf{j}+\frac{1}{2}\mathbf{e}^n} \subset \partial\Omega$.

[7] If ∇p is needed, $\mathbf{a}^*$ will have to be extrapolated by (56) to the boundary to obtain the Neumann boundary condition for p.

[8] The main arguments are (i) the velocity is at best fourth-order accurate, (ii) the Neumann boundary condition of ∇q in (34d) cannot be calculated to fourth-order accurate near the boundary due to the lop-sided ghost-filling formulas in Sect. 3.3 and the $O(1/h)$ coefficients in (44) and (45), and (iii) the error in calculating the boundary condition causes an error of the same order in the solution, since the Green's function defined by $G''(x) = 0,\ x \in [0,1];\ G'(0) = 1,\ G(1) = 0$ is $G(x) = x - 1$.

As for (76c), note that $\nabla \cdot \mathbf{u} = 0$ on $\partial\Omega$ should not be listed as a boundary condition in (34) because it can be deduced from the GePUP governing equations. However, from an *algorithmic* viewpoint, the condition (76c) is needed for pressure extraction from the velocity; see Sect. 3.4 for the details on how to utilize (74b), (76b), and (76c) to solve (74b).

As mentioned in Sect. 3.2, $\mathbf{G}$, $\mathbf{D}$, and $\mathbf{L}$ are fourth-order accurate in approximating their continuous counterparts ∇, $\nabla\cdot$, and Δ, respectively. By Sect. 4.1, $\mathbf{P}$ is also a fourth-order approximation of the Leray–Helmholtz projection $\mathscr{P}$. Therefore, the ODE system (74), (75), and (76) are formally a fourth-order discretization of the governing equations (34) and (35).

For time-dependent boundary conditions, it is well-known that Runge–Kutta methods applied to ODEs from the method-of-lines discretization could lead to order reduction in the solutions at the final times. Fortunately, on static no-slip domains, the boundary value of the velocity is a constant vector zero, and the analyses in [3,44] imply that no order reduction could come from the coupling of Runge–Kutta methods and method-of-lines discretization.

Away from the domain boundary, the operators $\mathbf{G}$, $\mathbf{D}$, and $\mathbf{L}$ have fourth-order truncation errors, thanks to their symmetric stencils of discretization. However, due to the fifth-order extrapolation formulas (53), the discrete Laplacian $\mathbf{L}$ in (46) has third-order truncation errors near the domain boundary. By the maximum principle, the adverse effect of these large truncation errors on the final solution may be negligible if the grid size and the time step size are small enough. As a *caveat*, the discrete system (74) is *nonlinear* and the discrete convection could expand the spectrum of the system by producing new high-frequency modes from existing low-frequency modes. In addition, the arguments in [62, Sect. 2] suggest that the corresponding small grid size and time-step size might not be affordable to recover the full fourth-order accuracy for a very small viscosity; after all, the computational resource is always limited in the real world. This partly explains why ghost cells are filled in Sect. 3.3 by fifth-order formulas: they reduce the truncation errors as much as possible.

5.1 GePUP-IMEX: Semi-implicit Time Stepping for the INSE

The RHS of (74a) can be split into two groups:

$$\mathbf{X}^{[\mathrm{E}]} := \langle \mathbf{g} \rangle - \mathbf{D}\,\langle \mathbf{uu} \rangle - \mathbf{G}\,\langle q \rangle\,, \quad \mathbf{X}^{[\mathrm{I}]} := \nu \mathbf{L}\,\langle \mathbf{w} \rangle\,, \tag{77}$$

where $\mathbf{L}$ applies component-wise to $\langle \mathbf{w} \rangle$ while computing a component of the convection term involves all those of the velocity vector.

For flows with low Reynolds number, it is desirable to treat the convection term explicitly and the diffusion term implicitly, because implicit treatment of the non-stiff convection term leads to a linear system challenging for iterative solvers whereas explicit treatment of the stiff diffusion term incurs very small time steps. For this purpose, an implicit-explicit (IMEX) scheme [4] can be coupled to GePUP in a straightforward way to form GePUP-IMEX, a group of semi-implicit time-marching schemes for the INSE. In this work, the ERK-ESDIRK method by Kennedy and Carpenter [27] is adopted as an example of GePUP-IMEX. For an ODE of the form

$$\frac{\mathrm{d}\psi}{\mathrm{d}t} = \mathbf{X}^{[\mathrm{E}]}(\psi, t) + \mathbf{X}^{[\mathrm{I}]}(\psi), \tag{78}$$

the steps of the aforementioned ERK-ESDIRK method are

$$\psi^{(1)} = \psi^n \approx \psi(t^n), \tag{79a}$$

$$\text{for } s = 2, 3, \ldots, n_{\mathrm{s}},$$

$$\left(I - \Delta t \gamma \mathbf{X}^{[\mathrm{I}]}\right)\psi^{(s)} = \psi^n + \Delta t \sum_{j=1}^{s-1} a_{s,j}^{[E]} \mathbf{X}^{[\mathrm{E}]}\left(\psi^{(j)}, t^{(j)}\right) + \Delta t \sum_{j=1}^{s-1} a_{s,j}^{[I]} \mathbf{X}^{[\mathrm{I}]}\psi^{(j)}, \tag{79b}$$

$$\psi^{n+1} = \psi^{(n_{\mathrm{s}})} + \Delta t \sum_{j=1}^{n_{\mathrm{s}}} \left(b_j - a_{n_{\mathrm{s}},j}^{[E]}\right) \mathbf{X}^{[\mathrm{E}]}\left(\psi^{(j)}, t^{(j)}\right), \tag{79c}$$

where the superscript $^{(s)}$ denotes an intermediate stage, $t^{(s)} = t^n + c_s \Delta t$ the time of that stage, n_{s} the number of stages, and A, $\mathbf{b}$, $\mathbf{c}$ the standard coefficients of the *Butcher tableau*. $A^{[E]}$, $\mathbf{b}$, $\mathbf{c}$ represent an ERK method while $A^{[I]}$, $\mathbf{b}$, $\mathbf{c}$ an ESDIRK method; see [27] for more details. At each intermediate stage, $\psi^{(s)}$ is determined by solving the linear system (79b) with its RHS calculated from previous stages, hence (79) is indeed *semi-implicit*. Note that (79c) follows from $b_j^{[E]} = b_j^{[I]} = b_j = a_{n_{\mathrm{s}},j}^{[I]}$, a simplification of the order conditions in deriving the ERK-ESDIRK method.

Applying the ERK-ESDIRK method (79) to the ODE system (74) and its IMEX splitting (77) yields the GePUP-IMEX algorithm as follows:

$$\langle \mathbf{w} \rangle^{(1)} = \langle \mathbf{w} \rangle^n, \tag{80a}$$

$$\begin{cases} \text{for } s = 2, 3, \ldots, n_{\mathrm{s}}, \\ \left(\mathbf{I} - \Delta t \nu \gamma \mathbf{L}\right) \langle \mathbf{w} \rangle^{(s)} = \langle \mathbf{w} \rangle^n + \Delta t \sum_{j=1}^{s-1} a_{s,j}^{[\mathrm{E}]} \mathbf{X}^{[\mathrm{E}]}\left(\langle \mathbf{u} \rangle^{(j)}, t^{(j)}\right) \\ \qquad\qquad + \Delta t \nu \sum_{j=1}^{s-1} a_{s,j}^{[\mathrm{I}]} \mathbf{L} \langle \mathbf{w} \rangle^{(j)}, \\ \langle \mathbf{u} \rangle^{(s)} = \mathbf{P} \langle \mathbf{w} \rangle^{(s)}, \end{cases} \tag{80b}$$

$$\begin{cases} \langle \mathbf{w} \rangle^* = \langle \mathbf{w} \rangle^{(n_{\mathrm{s}})} + \Delta t \sum_{j=1}^{n_{\mathrm{s}}} \left(b_j - a_{n_{\mathrm{s}},j}^{[E]}\right) \mathbf{X}^{[\mathrm{E}]}\left(\langle \mathbf{u} \rangle^{(j)}, t^{(j)}\right), \\ \langle \mathbf{u} \rangle^{n+1} = \mathbf{P} \langle \mathbf{w} \rangle^*, \\ \langle \mathbf{w} \rangle^{n+1} = \langle \mathbf{u} \rangle^{n+1}, \end{cases} \tag{80c}$$

where $\langle \mathbf{w} \rangle^0 = \langle \mathbf{u} \rangle^0 \approx \langle \mathbf{u}(t_0) \rangle$ at the initial time $t = t_0$ and $\langle \mathbf{w} \rangle^{t_e/\Delta t}$ is taken as the solution at the final time $t = t_e$.

The GePUP-IMEX algorithm (80) directly results from a *seamless* coupling of ERK-ESDIRK and the discrete GePUP formulation (74). In (80b), the first step corresponds to (74a). To prepare the evaluation of $\mathbf{X}^{[\mathrm{E}]}(\langle \mathbf{u} \rangle^{(j)}, t^{(j)})$ in the next stage, the discrete projection is applied to $\langle \mathbf{w} \rangle^{(s)}$ to obtain $\langle \mathbf{u} \rangle^{(s)}$. Due to the nonlinear convection term in $\mathbf{X}^{[\mathrm{E}]}$, the divergence of the vector $\langle \mathbf{w} \rangle^*$ in (80c) is much greater than that of $\langle \mathbf{w} \rangle^{(s)}$ in previous stages. Hence, the final solution $\langle \mathbf{w} \rangle^{n+1}$ of this time step is set to $\langle \mathbf{u} \rangle^{n+1}$, not $\langle \mathbf{w} \rangle^*$. This is also supported by the discussion in the second paragraph of Sect. 2.5.3 that $\mathbf{w}$ and $\mathbf{u}$ are consistent.

As evident from (80b) and (74), three Helmholtz-like linear systems have to be solved within each intermediate stage:

(HLS-1) the discrete Poisson's equation (74b) with the nonhomogeneous Neumann condition (76b),

(HLS-2) the Helmholtz-like linear system in (80b) with no-slip boundary conditions, i.e. $\mathbf{u}^{(j)} = \mathbf{0}$ and $\mathbf{w}^{(j)} = \mathbf{0}$ on $\partial\Omega$,

(HLS-3) the discrete Poisson's equation (66) with homogeneous Neumann conditions for projecting $\langle \mathbf{w} \rangle^{(s)}$ and $\langle \mathbf{w} \rangle^*$ onto the divergence-free space.

The algorithms for solving (HLS-1) and (HLS-3) have already been discussed in Sect. 3.4 and Sect. 4.1, respectively. For both BVPs, the unique solution is selected by requiring its integral over the domain be zero. In practice, the discrete solution of a purely homogeneous Neumann BVP is summed up, and then a constant is calculated and subtracted to enforce the zero sum of the solution.

In (80c), (HLS-1) and (HLS-3) have to be solved again, hence within each time step of GePUP-IMEX a total of 17 linear systems have to be solved. Standard multigrid V-cycles are used with pointwise weighted ($\omega = \frac{2}{3}$) Jacobi relaxation, injection, and constant restriction [10]. As confirmed by the numerical tests in Sect. 6, this works well for both the Laplacian operator and the Helmholtz-type operator. The reason is that $-\mathbf{L}$, the negation of the discrete Laplacian, is of the *essentially positive type* [9] [52, p. 439], which carries the same crucial properties of M-matrices such as positive definiteness. Consequently, the Helmholtz-type operator $\left(\mathbf{I} - \Delta t \nu \gamma \mathbf{L}\right)$ in (HLS-2) also has eigenvalues of positive real parts. Different from those of periodic domains, the smoothing operators change their forms near the domain boundary. In this case, care must be taken to derive the main diagonals of the Jacobi relaxation matrices from the stencil (46) and the ghost filling formulas (53). Numerical experiments show that this is crucial to obtain residual-reduction rates close to those in multigrid textbooks.

To estimate the range of stable Courant numbers, the convection term is first linearized, then a matrix counterpart of the approximate projection is assembled to verify that its spectral radius be one, as discussed in Sect. 4. Finally, the range of stable Courant numbers is obtained as the same with that of the advection-diffusion equation [60]:

$$\mathsf{Cr} \leq \frac{2.91}{\mathsf{D}}. \tag{81}$$

5.2 GePUP-ERK: Explicit Time Stepping for the INSE

For flows with high Reynolds number, an explicit Runge–Kutta (ERK) method may be more efficient than a semi-implicit method. The prominent advantage of the GePUP formulation that the velocity and the pressure are *completely* decoupled admits a straightforward formation of GePUP-ERK, a group of explicit time stepping schemes for the INSE as follows.

$$\langle \mathbf{w} \rangle^{(1)} = \langle \mathbf{w} \rangle^{n}, \tag{82a}$$

$$\begin{cases} \text{for } s = 2, 3, \ldots, n_{\mathsf{s}}, \\ \qquad \langle \mathbf{w} \rangle^{(s)} = \langle \mathbf{w} \rangle^{n} + \Delta t \sum_{j=1}^{s-1} a_{s,j}^{[\mathrm{E}]} \mathbf{Y}\left(\langle \mathbf{w} \rangle^{(j)}, \langle \mathbf{u} \rangle^{(j)}, t^{(j)}\right) \\ \qquad \langle \mathbf{u} \rangle^{(s)} = \mathbf{P} \langle \mathbf{w} \rangle^{(s)}, \end{cases} \tag{82b}$$

$$\begin{cases} \langle \mathbf{w} \rangle^{*} = \langle \mathbf{w} \rangle^{n} + \Delta t \sum_{j=1}^{n_{\mathsf{s}}} b_j \mathbf{Y}\left(\langle \mathbf{w} \rangle^{(j)}, \langle \mathbf{u} \rangle^{(j)}, t^{(j)}\right) \\ \langle \mathbf{u} \rangle^{n+1} = \mathbf{P} \langle \mathbf{w} \rangle^{*}, \\ \langle \mathbf{w} \rangle^{n+1} = \langle \mathbf{u} \rangle^{n+1}, \end{cases} \tag{82c}$$

where $t^{(j)} = t^n + c_j \Delta t$,

$$\mathbf{Y}\left(\langle \mathbf{w} \rangle^{(j)}, \langle \mathbf{u} \rangle^{(j)}, t^{(j)}\right) := \mathbf{X}^{[\mathrm{E}]}\left(\langle \mathbf{u} \rangle^{(j)}, t^{(j)}\right) + \nu \mathbf{L} \langle \mathbf{w} \rangle^{(j)}, \tag{83}$$

and $A^{[E]}$, $\mathbf{b}$, $\mathbf{c}$ denote an ERK method. For example, the classical fourth-order Runge–Kutta method is represented by $n_{\mathsf{s}} = 4$ and

$$\begin{array}{c|c} \mathbf{c} & A^{[E]} \\ \hline & \mathbf{b}^T \end{array} = \begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}. \tag{84}$$

As discussed in the previous subsection, overwriting $\langle \mathbf{w} \rangle^{n+1}$ with $\langle \mathbf{u} \rangle^{n+1}$ enforces the consistency between $\mathbf{w}$ and $\mathbf{u}$.

One might think that the GePUP-ERK algorithm (82) is much more efficient than the GePUP-IMEX algorithm (80) for high-Reynolds-number flows. However, timing results of numerical experiments show that GePUP-ERK is only about 1.5 times faster than GePUP-IMEX. This performance ratio is also supported by the following arguments. First, the only explicit step in the GePUP-ERK algorithm (82) is the computation of the predicted intermediate values $\langle \mathbf{w} \rangle^{(s)}$. For extracting the pressure gradient $\mathbf{G}\,\langle q \rangle$ and projecting $\langle \mathbf{w} \rangle^{(s)}$ to obtain the intermediate velocity, the two linear systems (HLS-1) and (HLS-3) still need to be solved. Second, the linear system (HLS-2) avoided in GePUP-ERK is the least time-consuming one among the three linear systems. As shown in Fig. 5, typically it only takes three multigrid v-cycles to solve (HLS-2); the reason is that a guess of the solution using that of the previous stage is already very close to the true solution. Consequently, the time consumed by the five multigrid solves of (HLS-2) within one time step of (80) is about the same as that of one (HLS-1) solve or one (HLS-3) solve. Third, although 17 linear systems have to be solved by GePUP-IMEX during each time step, the consumed CPU time is roughly equal to that of 13 (HLS-1) or (HLS-3) solves. Considering the fact that there are 8 implicit solves of either (HLS-1) or (HLS-3) in GePUP-ERK within each time step, the author estimates that GePUP-ERK is only 50 % faster than GePUP-IMEX, which is confirmed by the timing results of numerical experiments. Note that this advantage of GePUP-ERK may be lost for stiff problems, where numerical stability of GePUP-ERK dictates that the Courant number be in a range much narrower than (81).

In designing GePUP-ERK and GePUP-IMEX, the time integrator is treated as a black box, with its input being the temporal samples of the operators in (77) at intermediate stages, and its output being the solution of (74) at the end of a time step. Provided that the samples are accurate enough, the signature of the employed time integrator guarantees the accuracy. Apart from its simplicity, this black-box approach is attractive in that the internal details of the ODE solver is *completely* decoupled from time-stepping schemes for the INSE. As a benefit, it is straightforward to switch the time-marching scheme to others[9] such as those in [8,26,38]. In this sense, each of GePUP-ERK and GePUP-IMEX is more of a group of high-order schemes than merely a single method.

Finally, we end this section by emphasizing that *GePUP-ERK and GePUP-IMEX are fourth-order accurate both in space and in time* because

- the spatial discretization of the GePUP governing equations (34) and (35) into the ODE system (74) is fourth-order accurate,
- a standard time integrator applies directly to the discrete ODE system (74), of which the boundary conditions have nothing to do with the time integrator,
- the time integrator applied to (74) is fourth-order accurate.

6 Numerical Tests

In this section, the discrete projection $\mathbf{P}$ discussed in Sect. 4 is first shown to be fourth-order accurate. As a side evidence of its unit spectral radius, repeated application of $\mathbf{P}$ to a discrete vector field converges. Benchmark tests are then performed both in two dimensions (2D) and in three dimensions (3D) to demonstrate the fourth-order accuracy of the GePUP schemes for various Reynolds numbers, Courant numbers, and grid sizes. An order reduction at the domain corners is also observed and discussed. In the rest of this paper, all presented

[9] Two motivations for doing so could be an even higher temporal accuracy or a better stability for very stiff problems.

Table 1 Errors and convergence rates of the discrete projection **P** acting on $\mathbf{u}^* = \mathbf{u} + \nabla\phi$ with **u** and ϕ defined in (85a) and (86), respectively

h	$\frac{1}{64}$	Rate	$\frac{1}{128}$	Rate	$\frac{1}{256}$
$\mathbf{Pu}^*\ L_\infty$	2.02e−06	3.96	1.30e−07	3.99	8.20e−09
$\phi\ L_\infty$	6.47e−08	4.01	4.01e−09	4.00	2.50e−10
$\nabla\cdot\mathbf{Pu}^*\ L_\infty$	1.39e−04	2.96	1.79e−05	2.99	2.25e−06
$\nabla\cdot\mathbf{Pu}^*\ L_1$	1.39e−05	3.94	9.02e−07	3.99	5.70e−08
$\nabla\cdot\mathbf{Pu}^*\ L_2$	2.76e−05	3.49	2.46e−06	3.50	2.17e−07
$\nabla\nabla\cdot\mathbf{Pu}^*\ L_\infty$	1.31e−02	1.97	3.34e−03	1.99	8.38e−04
$\nabla\nabla\cdot\mathbf{Pu}^*\ L_1$	5.18e−04	3.02	6.40e−05	3.03	7.86e−06
$\nabla\nabla\cdot\mathbf{Pu}^*\ L_2$	1.83e−03	2.47	3.31e−04	2.49	5.88e−05
$\lvert\langle\mathbf{Pu}^*, \nabla\phi\rangle_V\rvert$	1.08e−07	4.04	6.54e−09	4.03	4.01e−10

Results of the 3D test with (85b) and (86) are quantitatively the same with slightly smaller errors

numerical results are those of the GePUP-IMEX algorithm. For all tests with $\mathrm{Re} \geq 10^4$, the GePUP-ERK scheme is also performed with the classical Runge–Kutta method (84) and the results differ from those of GePUP-IMEX by at most a half of one percent.

6.1 Testing the Discrete Projection

In this test, a vector field $\mathbf{u}^* = \mathbf{u} + \nabla\phi$ is defined by

$$\mathbf{u}(x, y) = \begin{pmatrix} \sin^2(\pi x)\,\sin(2\pi y) \\ -\sin(2\pi x)\,\sin^2(\pi y) \end{pmatrix}, \tag{85a}$$

$$\mathbf{u}(x, y, z) = \frac{1}{2}\begin{pmatrix} \sin^2(\pi x)\,\sin(2\pi y)\,\sin(2\pi z) \\ \sin(2\pi x)\,\sin^2(\pi y)\,\sin(2\pi z) \\ -2\sin(2\pi x)\,\sin(2\pi y)\,\sin^2(\pi z) \end{pmatrix}, \tag{85b}$$

$$\phi(\mathbf{x}) = \Pi_{d=1}^{\mathrm{D}} \sin(2\pi x_d), \tag{86}$$

where **u** is clearly solenoidal both in 2D and 3D.

The Simpson's $\frac{3}{8}$ rule is used to compute the cell averages $\langle\mathbf{u}^*\rangle$, to which the discrete projection **P** is applied *once*. The errors from the cell-averaged exact solutions and the corresponding convergence rates are then calculated. As shown in Table 1, both the projected vector $\mathbf{Pu}^*$ and the scalar ϕ are computed to fourth-order accuracy. By (52) and the last row of Table 1, the L_2 orthogonality of $\mathbf{Pu}^*$ and $\nabla\phi$ is also fulfilled to fourth-order accuracy. The convergence rate of the divergence of the projected vector is 3 in the max-norm, 4 in the 1-norm, and 3.5 in the 2-norm, implying[10] that the order reduction is on a set of co-dimension one. Error plots (not shown here) indicate that the cells of this co-dimension-one set are along the domain boundary. This order reduction is due to the fourth-order accuracy

[10] Following [32, Sec. A.5], the accuracy of numerical results are measured by the p-norm of a grid function as $\|\psi\|_p = \left(h^{\mathrm{D}}\sum_{\mathbf{i}\subseteq\Omega}|\psi_{\mathbf{i}}|^p\right)^{1/p}$. If $\psi = O(h^3)$ on $O(\frac{1}{h^{\mathrm{D}-1}})$ cells and $\psi = O(h^4)$ on $O(\frac{1}{h^{\mathrm{D}}})$ cells, then $\|\psi\|_1 = h^{\mathrm{D}}O(h^4)O(\frac{1}{h^{\mathrm{D}}}) + h^{\mathrm{D}}O(h^3)O(\frac{1}{h^{\mathrm{D}-1}}) = O(h^4)$ and $\|\psi\|_2 = \left(h^{\mathrm{D}}O(h^8)O(\frac{1}{h^{\mathrm{D}}}) + h^{\mathrm{D}}O(h^6)O(\frac{1}{h^{\mathrm{D}-1}})\right)^{1/2} = O(h^{3.5})$.

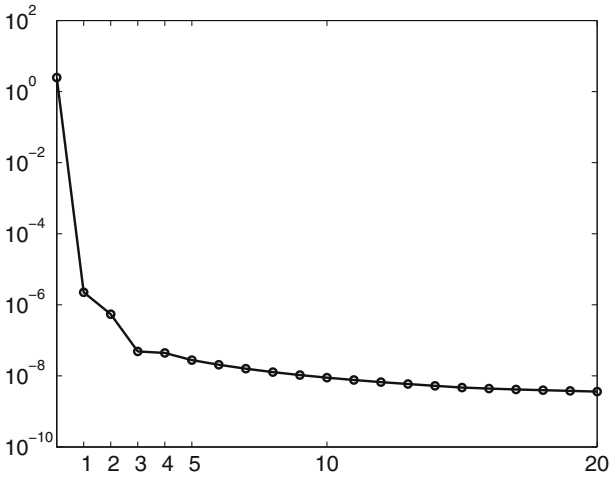

Fig. 3 The max-norm of the divergence of the projected velocity decreases monotonically with the number of projections on a 2D grid ($h = \frac{1}{256}$). The abscissa is n, the number of repeated projections, and the ordinate is $\|\mathbf{DP}^n \langle \mathbf{u}^* \rangle\|_\infty$. Results of the 3D test are qualitatively the same

of the cell-averaged velocity, the lop-sided ghost filling formulas (53), and the coefficient $\frac{1}{h}$ in (45). Similarly, the convergence rates of $\nabla\nabla \cdot \mathbf{Pu}^*$ are less than those of $\nabla \cdot \mathbf{Pu}^*$ by one for all norms.

On the finest grid, the discrete projection $\mathbf{P}$ is repeatedly applied to $\langle \mathbf{u}^* \rangle$. The max-norm of the divergence of the projected vector with respect to the number of projections is plotted in Fig. 3. The convergence of $\|\mathbf{DP}^n \langle \mathbf{u}^* \rangle\|_\infty$ supports the results in Sect. 4.2 that the spectral radius of $\mathbf{P}$ is one.

In another group of tests, the vector is set to $\mathbf{u}^* = \nabla\phi$ for examining the accuracy of the discrete projection $\mathbf{P}$ in annihilating gradient fields. The exact solution of the projected vector is clearly a constant zero vector. For these tests, the errors and convergence rates are almost the same as those in Table 1, with each pair of the corresponding numbers for these two cases differing at most one percent. This implies that $\mathbf{P}\langle \nabla\phi \rangle = O(h^4)$.

Due to the low accuracy of $\nabla\nabla \cdot \mathbf{Pu}^*$, the assumption (33a) might appear unwarranted. However, as discussed in Sect. 2.5.2, the maximum principle of the heat equation (36) ensures that $\nabla \cdot \mathbf{w}$ decays exponentially fast. Furthermore, a Fourier-series solution of the heat equation implies that $\nabla\nabla \cdot \mathbf{Pu}$ also decays exponentially fast. Hence, the inexactness of the discrete projection does not affect the convergence rates of the proposed INSE algorithms.

6.2 Single-Vortex Tests

Following [7], an axisymmetric velocity field is defined on the unit box $[0, 1]^2$ by

$$u_\theta(r_v) = \begin{cases} \Gamma(\frac{1}{2}r_v - 4r_v^3) & \text{if } r_v < R; \\ \Gamma\frac{R}{r_v}(\frac{1}{2}R - 4R^3) & \text{if } r_v \geq R, \end{cases} \tag{87}$$

where r_v is the radial distance from the vortex center $(\frac{1}{2}, \frac{1}{2})^T$. The choice of $R = 0.2$ and $\Gamma = 1.0$ leads to $\max(u_\theta) = 0.068$. A small viscosity $\nu = 3.4 \times 10^{-6}$ yields a high Reynolds number Re = 20,000. The initial condition of cell-averaged velocity is first calculated by (87)

and a sixth-order Newton–Cotes quadrature formula (Boole's rule); it is then projected by $\mathbf{P}$ ten times so that it is approximately divergence-free, c.f. Fig. 3.

Starting with an initial Courant number of 0.5, the time-step size is adaptively changed according to the velocity with a maximum increase rate of 1.1. Within fifty time steps, the Courant number Cr reaches its upper bound of 1.0, and the total number of time steps is roughly the same as that of running the test with the time-step size fixed at a constant Cr = 1.0. However, adaptive time-stepping at the beginning stage of a simulation is recommended for tests with large velocity divergence in the initial condition. As discussed in the precious section, the discrete velocity divergence is only third-order accurate near the domain boundary, which has an adverse effect on the extrapolation of $\frac{\partial q}{\partial n}$ from $\frac{\partial \nabla\cdot\mathbf{u}}{\partial n}$ as in (61). Fortunately, the larger number of adaptively changed time steps in the beginning stage leads to a stronger decay of $\nabla\cdot\mathbf{u}$, and thus reduces the unfavorable influence of $\nabla\cdot\mathbf{u}$. For this purpose, one-step methods are advantageous over multistep methods in that their ease of changing the time-step size fits nicely in the whole scenario.

The tests are performed on four successively refined uniform grids. The time span [0, 60] is made long enough for the turbulent boundary layers to develop prominent Lagrangian coherent structures. Figure 4 shows snapshots of the vorticity at time $t = 40$ and at the final time $t = 60$. The essential features of vortex sheet roll-up and counter-vortices agree with those in [7].

Since no exact solutions are available, the error on a grid is defined by standard Richardson extrapolation, i.e. it is the difference of solutions on the current grid and the next coarser grid. As shown in Table 2, the convergence rates of the velocity and the pressure p in all norms have order reductions on coarse grids, but they grow to almost four as the grids are refined. These results confirm the discussion at the beginning of Sect. 5: although large truncation errors near the domain boundary could lead to order reductions of the velocity, its fourth-order accuracy may be ubiquitously recovered if the grid size and time step size are small enough. In comparison, the convergence rates of vorticity and velocity divergence are around three in the max-norm, because of the fourth-order accuracy of the velocity, the lop-sided ghost-filling formulas (53) and the constant $\frac{1}{h}$ in (44).

Also shown in Table 2, the convergence rates of the scalar q obtained from PEV are around 2.5, less than those of the pressure p from PEEA. As discussed in Sect. 4.3, extrapolating the value of $\frac{\partial q}{\partial n}$ on the boundary via the formula (58) from cell-averaged velocity is only second-order accurate. Consequently, the pressure q obtained in Sect. 3.4 is also second-order accurate in the worst case. As shown in the last row of Table 2, the L_2 orthogonality of $\mathbf{u}$ and ∇q is fulfilled not only to the fourth-order accuracy, but also close to machine precision on the finest grid. Thanks to this L_2 orthogonality, the lower accuracy of ∇q does not affect the fourth-order accuracy of the velocity.

6.3 Viscous-Box Tests

Following [6], the initial velocity of this test is defined by (85). Note that both (85a) and (85b) agree with the structure of boundary layers in terms of the velocity gradient tensor, i.e. they both satisfy (62).

The cell-averaged velocity at the initial time is calculated by (85) and Boole's rule. It is then advanced from $t_0 = 0$ to $t_e = 0.5$ by fixing the time-step size on four successively refined uniform grids. As shown in Table 3, the convergence rates of both the velocity and the pressure p are very close to four in the case of $\mathrm{Re} = 10^4$; those of the vorticity are also satisfactory. The growth of the convergence rates is similar to that observed in Table 2,

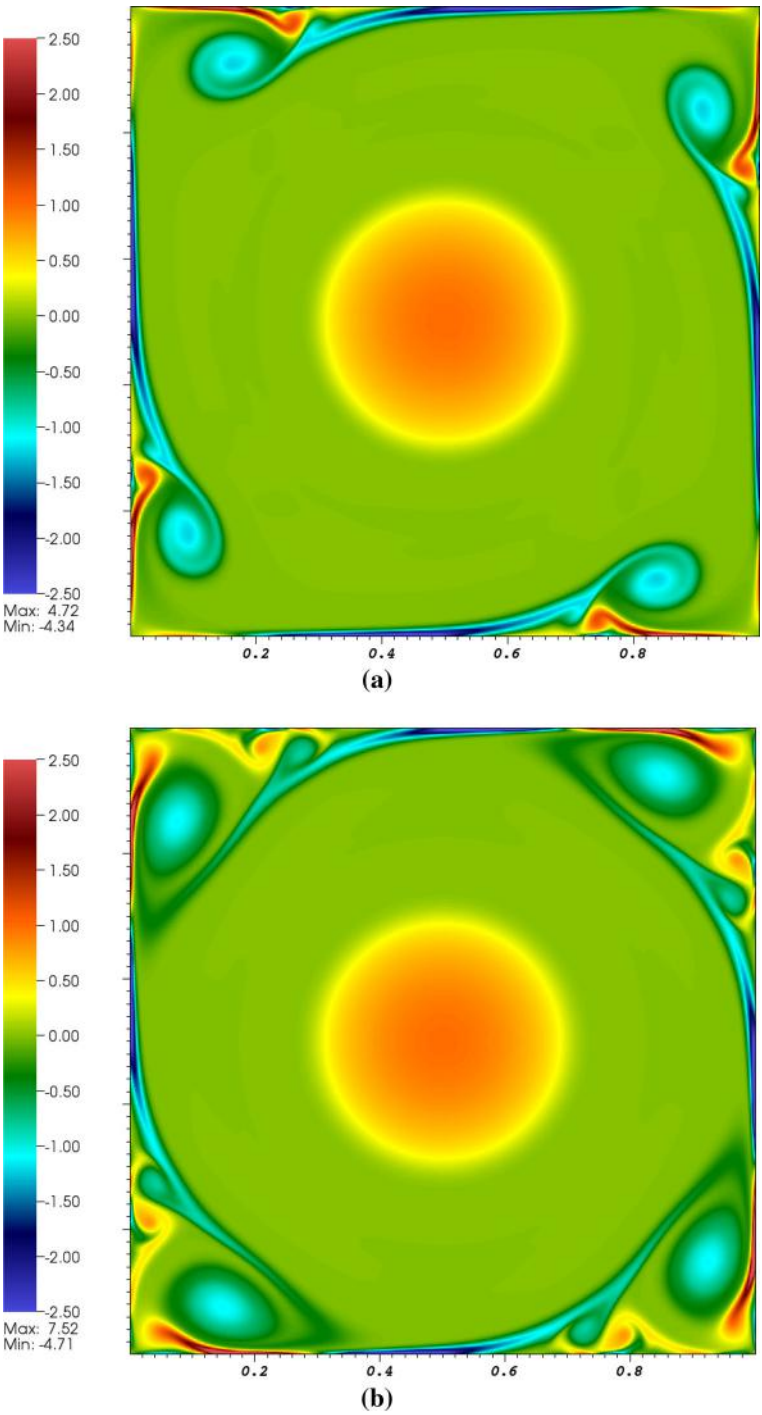

Fig. 4 Snapshots of vorticity for the single-vortex test on a uniform grid with $h = 1/1024$. The region of each cell is filled by a single color that corresponds to the cell-averaged vorticity. No image smoothing is applied. **a** t = 40. **b** t = 60 (Color figure online)

Table 2 Errors and convergence rates of GePUP-IMEX for the single-vortex test with $\mathrm{Re} = 20,000$, $t_0 = 0.0$, $t_e = 60$, adaptive time-stepping, and the terminal Courant number Cr = 1.0

h	$\frac{1}{256}-\frac{1}{512}$	Rate	$\frac{1}{512}-\frac{1}{1024}$	Rate	$\frac{1}{1024}-\frac{1}{2048}$
$\mathbf{u}\ L_\infty$	9.61e−04	3.66	7.60e−05	3.85	5.26e−06
$\mathbf{u}\ L_1$	3.92e−05	3.78	2.85e−06	3.90	1.91e−07
$\mathbf{u}\ L_2$	7.44e−05	3.76	5.51e−06	3.88	3.73e−07
$\nabla q\ L_\infty$	5.54e−04	2.32	1.11e−04	2.83	1.56e−05
$\nabla q\ L_1$	1.82e−05	3.24	1.92e−06	2.74	2.88e−07
$\nabla q\ L_2$	3.84e−05	3.15	4.32e−06	2.70	6.65e−07
$p\ L_\infty$	7.01e−06	3.78	5.09e−07	3.90	3.40e−08
$p\ L_1$	9.00e−07	3.81	6.41e−08	3.90	4.29e−09
$p\ L_2$	1.41e−06	3.80	1.01e−07	3.90	6.77e−09
$q\ L_\infty$	9.23e−06	2.06	2.22e−06	2.36	4.34e−07
$q\ L_1$	8.03e−07	2.32	1.60e−07	2.44	2.96e−08
$q\ L_2$	1.40e−06	2.60	2.31e−07	2.50	4.09e−08
$\nabla\cdot\mathbf{u}\ L_\infty$	1.04e−02	1.99	2.62e−03	2.84	3.64e−04
$\nabla\cdot\mathbf{u}\ L_1$	5.31e−05	3.36	5.16e−06	3.99	3.25e−07
$\nabla\cdot\mathbf{u}\ L_2$	3.15e−04	2.44	5.80e−05	3.30	5.87e−06
$\nabla\times\mathbf{u}\ L_\infty$	2.20e−01	2.75	3.26e−02	3.12	3.74e−03
$\nabla\times\mathbf{u}\ L_1$	3.90e−03	3.60	3.22e−04	3.85	2.24e−05
$\nabla\times\mathbf{u}\ L_2$	1.15e−02	3.29	1.17e−03	3.71	8.97e−05
h	$\frac{1}{512}$	Rate	$\frac{1}{1024}$	Rate	$\frac{1}{2048}$
$\lvert\langle\mathbf{u},\nabla q\rangle_V\rvert$	1.57e−12	4.52	6.87e−14	4.11	3.98332e−15

GePUP-ERK yields results that are quantitatively the same up to a half of 1 %, but uses only 68 % of the total running time of GePUP-IMEX on each grid

Table 3 Errors and convergence rates of GePUP-IMEX for the 2D viscous box test with $\mathrm{Re} = 10^4$, $t_0 = 0.0$, $t_e = 0.5$, and Cr=0.5

h	$\frac{1}{128}-\frac{1}{256}$	Rate	$\frac{1}{256}-\frac{1}{512}$	Rate	$\frac{1}{512}-\frac{1}{1024}$
$\mathbf{u}\ L_\infty$	1.02e−03	3.20	1.10e−04	4.09	6.49e−06
$\mathbf{u}\ L_1$	4.12e−05	3.53	3.56e−06	3.94	2.32e−07
$\mathbf{u}\ L_2$	9.86e−05	3.50	8.71e−06	3.96	5.58e−07
$\nabla q\ L_\infty$	5.06e−03	1.61	1.66e−03	1.87	4.55e−04
$\nabla q\ L_1$	6.63e−04	1.56	2.25e−04	1.97	5.75e−05
$\nabla q\ L_2$	1.00e−03	1.60	3.31e−04	2.01	8.25e−05
$p\ L_2$	1.87e−05	3.33	1.86e−06	3.83	1.31e−07
$q\ L_2$	9.49e−05	1.13	4.34e−05	1.91	1.15e−05
$\nabla\times\mathbf{u}\ L_\infty$	3.21e−01	2.49	5.72e−02	3.62	4.64e−03
$\nabla\times\mathbf{u}\ L_2$	4.15e−02	2.80	5.96e−03	3.75	4.42e−04

The values of $\left|\langle\mathbf{u},\nabla q\rangle_V\right|$ on these four grids are 8.35e−9, 8.77e−10, 8.92e−11, and 3.54e−12; the corresponding convergence rates are 3.25, 3.89, and 4.06

confirming again that the large truncation errors near the no-slip boundaries are negligible if the grids are fine enough. The convergence rates of ∇q are around two in all norms. As mentioned in the caption of Table 3, convergence rates of fulfilling the L_2-orthogonality of $\mathbf{u}$ and ∇q also grow to four as the grids are refined.

Table 4 Errors and convergence rates of GePUP-IMEX for the 2D viscous box test with Re = 100, $t_0 = 0.0$, $t_e = 0.5$, and Cr=0.1

h	$\frac{1}{64}-\frac{1}{128}$	Rate	$\frac{1}{128}-\frac{1}{256}$	Rate	$\frac{1}{256}-\frac{1}{512}$
$\mathbf{u}\ L_\infty$	7.84e−06	2.48	1.41e−06	2.50	2.50e−07
$\mathbf{u}\ L_1$	2.08e−06	3.87	1.42e−07	3.84	9.95e−09
$\mathbf{u}\ L_2$	2.68e−06	3.86	1.84e−07	3.80	1.33e−08
$\nabla q\ L_\infty$	4.82e−03	0.76	2.85e−03	0.51	2.00e−03
$\nabla q\ L_1$	3.86e−04	2.15	8.73e−05	2.18	1.93e−05
$\nabla q\ L_2$	5.93e−04	1.87	1.62e−04	1.84	4.54e−05
$p\ L_2$	3.15e−06	3.13	3.60e−07	2.09	8.48e−08
$q\ L_2$	7.30e−05	2.15	1.64e−05	2.00	4.10e−06
$\nabla \times \mathbf{u}\ L_\infty$	1.52e−03	1.67	4.78e−04	2.16	1.07e−04
$\nabla \times \mathbf{u}\ L_2$	1.34e−04	3.06	1.60e−05	3.09	1.88e−06

The values of $|\langle \mathbf{u}, \nabla q\rangle_V|$ on these four grids are 3.77e−8, 2.83e−9, 1.92e−10, and 1.25e−11; the corresponding convergence rates are 3.73, 3.89, and 3.94

A smaller Courant number Re = 100 is also used for the viscous-box test. As shown in Table 4, the convergence rates of the velocity in this case are around 2.5 in the max-norm, and close to four both in the 1-norm and in the 2-norm. This order reduction is not due to the large truncation errors near the domain boundary because fourth-order convergence rates have already been achieved on the same grids in the case of Re = 10^4 and a larger viscosity implies a faster decaying rate of the truncation errors. Error plots (not shown here) indicate that the large errors of velocity are all concentrated in the regions near the four domain corners; this feature is also different from the order-reduction caused by the projection, as discussed in footnote 10 in Sect. 6.1. Interestingly, the errors of ∇q are closely correlated to those of $\mathbf{u}$. In other words, ∇q has the same error pattern with $\mathbf{u}$ in that its large errors are also concentrated at the domain corners, which is completely different from that for Re = 10^4. In Table 4, although the L_2-orthogonality of ∇q and $\mathbf{u}$ are fulfilled close to the fourth-order accuracy, the L_∞ convergence rates of the velocity do not seem to benefit from it.

As discussed in Sect. 2.2, the pressure gradient in the INSE can be split into two parts $\nabla p = \nabla p_c + \nu \nabla p_s$, where ∇p_s responses to the Laplace–Leray commutator. When ν is sufficiently large, $\nu \nabla p_s$ dominates ∇p_c and accounts for the bulk of ∇p. Cozzi and Pego [14] showed that $\|\nabla p_s\|$ may not be bounded at a boundary point that is not C^3. As a practical interpretation, the pressure could develop steep gradient at a C^1 discontinuity of the domain boundary for flows with low Reynolds number.

In light of the previous paragraph, the author speculates that the order reduction in the case of Re = 100 is not caused by defects of the proposed method, but by the dominance of $\nu \nabla p_s$ and the inevitable sharp corners (C^1 discontinuities) of rectangular domains. As a side evidence, ∇q and $\mathbf{u}$ in Table 3 are ubiquitously second-order accurate and fourth-order accurate, respectively, thanks to the dominance of ∇p_c that follows from a much smaller viscosity. Practically speaking, this order reduction for Re = 100 seems to be minor since the L_∞ convergence rate of the velocity is still well over two on the finest grid.

The same tests of Re = 10^4 and Re = 100 are also performed in 3D, with the results shown in Tables 5 and 6. Convergence rates of the velocity and the pressure in Table 5 are close to four while the results in Table 6 are qualitatively the same as those in 2D. The order reduction of the velocity appears to be more prominent; after all, the number of cells that abut a 3D C^1 discontinuity of $\partial\Omega$ is much larger than that of the 2D tests.

Figure 5 illustrates typical multigrid performance for various Reynolds numbers, Courant numbers, and grid sizes. In solving (HLS-1) to extract pressure q from the velocity, the

Table 5 Errors and convergence rates of GePUP-IMEX for the 3D viscous box test with $\mathrm{Re}=10^4$, $t_0 = 0.0$, $t_e = 0.5$, and $\mathrm{Cr}=0.5$

GePUP-ERK yields results that are quantitatively the same as follows up to a half of one percent, but uses only 65 % of the total running time of GePUP-IMEX on each grid

h	$\frac{1}{32}-\frac{1}{64}$	Rate	$\frac{1}{64}-\frac{1}{128}$	Rate	$\frac{1}{128}-\frac{1}{256}$
$\mathbf{u}\ L_\infty$	2.46e−02	0.97	1.26e−02	3.57	1.06e−03
$\mathbf{u}\ L_1$	1.20e−03	1.30	4.88e−04	3.21	5.25e−05
$\mathbf{u}\ L_2$	2.22e−03	1.10	1.04e−03	3.41	9.72e−05
$\nabla q\ L_\infty$	7.77e−02	0.62	5.06e−02	1.66	1.60e−02
$\nabla q\ L_1$	6.42e−03	0.93	3.38e−03	1.14	1.54e−03
$\nabla q\ L_2$	1.05e−02	0.67	6.57e−03	1.52	2.30e−03
$p\ L_\infty$	5.61e−03	1.02	2.77e−03	3.63	2.23e−04
$q\ L_\infty$	7.65e−03	0.18	6.74e−03	2.09	1.59e−03
$\nabla \times \mathbf{u}\ L_2$	3.37e−01	0.99	1.70e−01	2.98	2.15e−02

Table 6 Errors and convergence rates of GePUP-IMEX for the 3D viscous box test with $\mathrm{Re} = 100$, $t_0 = 0.0$, $t_e = 0.5$, and $\mathrm{Cr}=0.5$

h	$\frac{1}{32}-\frac{1}{64}$	Rate	$\frac{1}{64}-\frac{1}{128}$	Rate	$\frac{1}{128}-\frac{1}{256}$
$\mathbf{u}\ L_\infty$	1.46e−04	2.22	3.14e−05	2.35	6.16e−06
$\mathbf{u}\ L_1$	2.47e−05	3.62	2.00e−06	3.75	1.49e−07
$\mathbf{u}\ L_2$	3.25e−05	3.54	2.79e−06	3.68	2.18e−07
$\nabla q\ L_\infty$	1.53e−02	1.13	6.99e−03	0.86	3.85e−03
$\nabla q\ L_1$	1.74e−03	2.07	4.15e−04	2.28	8.55e−05
$\nabla q\ L_2$	2.69e−03	1.99	6.78e−04	2.10	1.58e−04
$p\ L_2$	3.87e−05	1.91	1.03e−05	2.31	2.08e−06
$q\ L_2$	3.72e−04	2.08	8.79e−05	2.40	1.67e−05
$\nabla \times \mathbf{u}\ L_2$	1.39e−03	2.67	2.18e−04	2.90	2.92e−05

initial residual is the largest among the three linear solves, because of the evaluation of the divergence of convection and the extrapolation of the velocity derivatives to the domain boundary. Estimating the projected velocity from (HLS-2) is the fastest among the three linear solves, since velocity of the previous stage is already a very good initial guess; this is especially true in the case of high Reynolds number and/or small Courant number. Indeed, for a small viscosity the linear system (HLS-2) has strong diagonal dominance and the weighted Jacobi is very effective in damping the high-frequency modes. In solving (HLS-3), the initial divergence error in projecting $\langle \mathbf{w} \rangle$ onto the divergence-free space is also very small, due to the fact that the non-solenoidal part of $\langle \mathbf{w} \rangle$ is proportional to the time step size. The multigrid performance in 3D tests are slightly worse than that of the corresponding 2D tests. This is not surprising since a wider spectrum of high-frequency modes exists in 3D.

Finally, the author remarks that the multigrid conditioning[11] for a 2048^2 grid is around 10^6 and that the initial residual of the three multigrid solves could differ five orders of magnitudes even for the non-stiff tests of high Reynolds number. In this sense, the proposed fourth-order method makes full use of the significands of the double-precision floating point numbers!

[11] The thresholds of relative convergence in Fig. 5 are set according to the condition numbers of multigrid solvers.

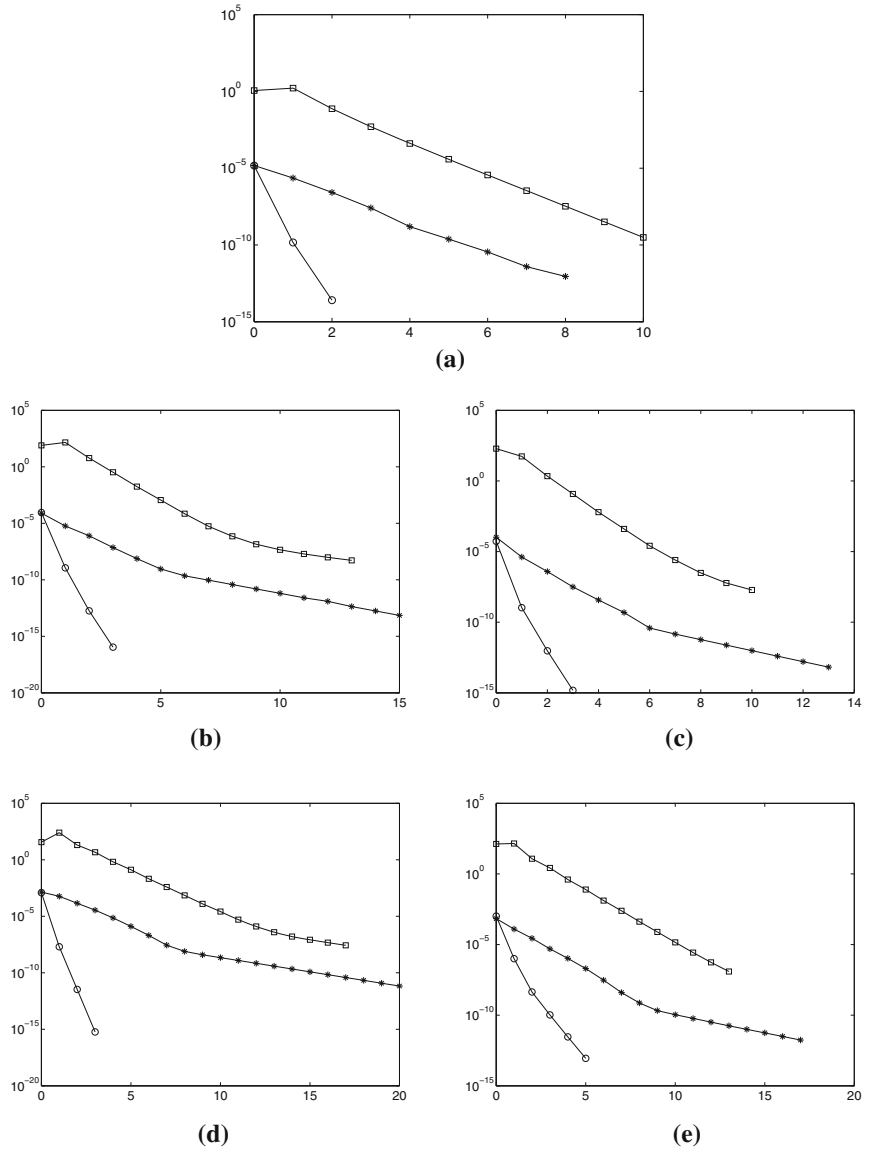

Fig. 5 Typical performance of multigrid V-cycles of GePUP-IMEX sampled at the middle of the tests. The abscissa and the ordinate are the iteration number of multigrid V-cycles and the max-norm of the residual of the solution, respectively. The *dots*, "*square*", "*circles*", and "*asterisk*" represent the residuals of the three linear systems (HLS-1), (HLS-2), and (HLS-3), i.e. those for extracting pressure, solving the Helmholtz-like system for the intermediate velocity, and enforcing incompressibility, respectively. Both pre-smoothing and post-smoothing contain four rounds of weighted Jacobi relaxation with the weight set to $\frac{2}{3}$. Injection and constant restriction are employed as intergrid transfer operators. For (**b**) and (**c**), the threshold of relative convergence is set to 10^{-10}; the multigrid solvers also stop when the absolute value of the residual is smaller than 10^{-13}. The tests of (**a**), (**d**), and (**e**) have larger conditions numbers; the relative and absolute convergence threshold are set to 10^{-9} and 10^{-12}, respectively. **a** Single-vortex: $\mathrm{Re}=2\times10^4$, $\mathrm{Cr}=1.0$, $h=\frac{1}{2048}$. **b** 2D Viscous-box: $\mathrm{Re}=10^4$, Cr=0.5, and $h=\frac{1}{1024}$. **c** 2D Viscous-box: Re = 100, Cr = 0.1, and $h=\frac{1}{512}$. **d** 3D Viscous-box: $\mathrm{Re}=10^4$, Cr = 0.5, and $h=\frac{1}{256}$. **e** 3D Viscous-box: R = 100, Cr = 0.5, and $h=\frac{1}{256}$

6.4 A 3D Lid-Driven Cavity

Incompressible flows inside a 3D lid-driven cavity are of great scientific interest because they exhibit almost all fluid mechanical phenomena: eddies, secondary flows, chaotic particle motions, instabilities, transition, turbulence, and complex 3D flow patterns such as spanwise vortices; see [47] for an overview. Also because of its extremely simple geometrical setting, the lid-driven cavity is one of the most popular numerical benchmarks for INSE solvers.

In this subsection, the proposed GePUP-IMEX algorithm is used to examine the start-up flows in a 3D rectangular driven cavity of aspect ratio 1:1:2 at Re = 1000. The computational domain is $x \times y \times z \in [0, 1] \times [0, 1] \times [0, 2]$. The boundary condition of the velocity $\mathbf{u} = (u, v, w)$ is $\mathbf{u} = \mathbf{0}$ for all boundaries except that $v = 1$ at $x = 1$. The initial condition is $\mathbf{u} = \mathbf{0}$ everywhere inside the domain. This computational setup exactly follows that of [19, Fig. 2], except that the z-span of the domain is shifted by 1 so that the spanwise symmetry plane of this work is at $z = 1$ instead of $z = 0$.

The computational domain is discretized by a uniform grid with $h = \frac{1}{128}$. A steady Courant number 0.5 is adopted to advance the solution from $t = 0$ to $t = 12$. The whole simulation of 3072 time steps over 4.2 million control volumes took 45 h on the author's personal desktop with a single Intel® i7-3930k CPU. As a well-known fact, the velocity component v has singularities at the lid, where the convergence rates of a numerical solver deteriorate. Since the fourth-order accuracy of the proposed INSE solvers have already been demonstrated by the previous two subsections, we focus on the comparison of current computational results to those of previous experiments and computations.

Streamline plots of the computed velocity field inside the symmetry plane at time $t = 2, 4, 6, 8, 10, 12$ are plotted in Fig. 6. Right after the simulation starts, the upward motion of the lid generates high shear stress along the cavity lid. By $t = 2$, an eddy has formed: the speed at the cell deemed at the center of the eddy is much smaller than those of all nearby cells by a factor of about 50. As the simulation continues, the high velocity fluid of this counter-clockwise-rotating eddy penetrates the initially quiescent fluid. Shortly after $t = 4$, an additional clockwise-rotating eddy starts to form and becomes eminent at $t = 6$. Then this secondary eddy intensifies and moves to the upper left corner in subsequent times. At time $t = 12$, the primary eddy, the secondary eddy, and the velocity field have reached a steady state in the sense that relative changes of the velocity field between two successive time steps are less than 10^{-4}. These qualitative features agree well with previous experimental results in [19, Fig. 6].

A major difference of the 3D lid-driven cavity flow from its 2D counterpart is the evolution of vortices at the spanwise wall ends and the Taylor-Görtler-like vortices near the symmetry plane. These prominent features are shown in Fig. 7 for three time instances.

Quantitative comparison of the proposed GePUP-IMEX algorithm with previous experiments and computation are presented in Fig. 8 for the center location of the primary eddy, and in Fig. 9 for the velocity profile inside two planes $z = 1, 0.5$. The comparison indicates excellent match of computational results between this work and that by Guermond et. al. [19]. The author's computational results also agree well with the experimental results in [19]. Furthermore, some results of this work, such as the v component of velocity in Fig. 9b, achieves a better agreement to experimental data than the computational results by Guermond et. al. [19].

To sum up, the author has demonstrated that the proposed INSE solvers based on the GePUP formulation are not only fourth-order accurate in time and in space, but also capable of capturing the right physics, i.e. *converging to the right solution*.

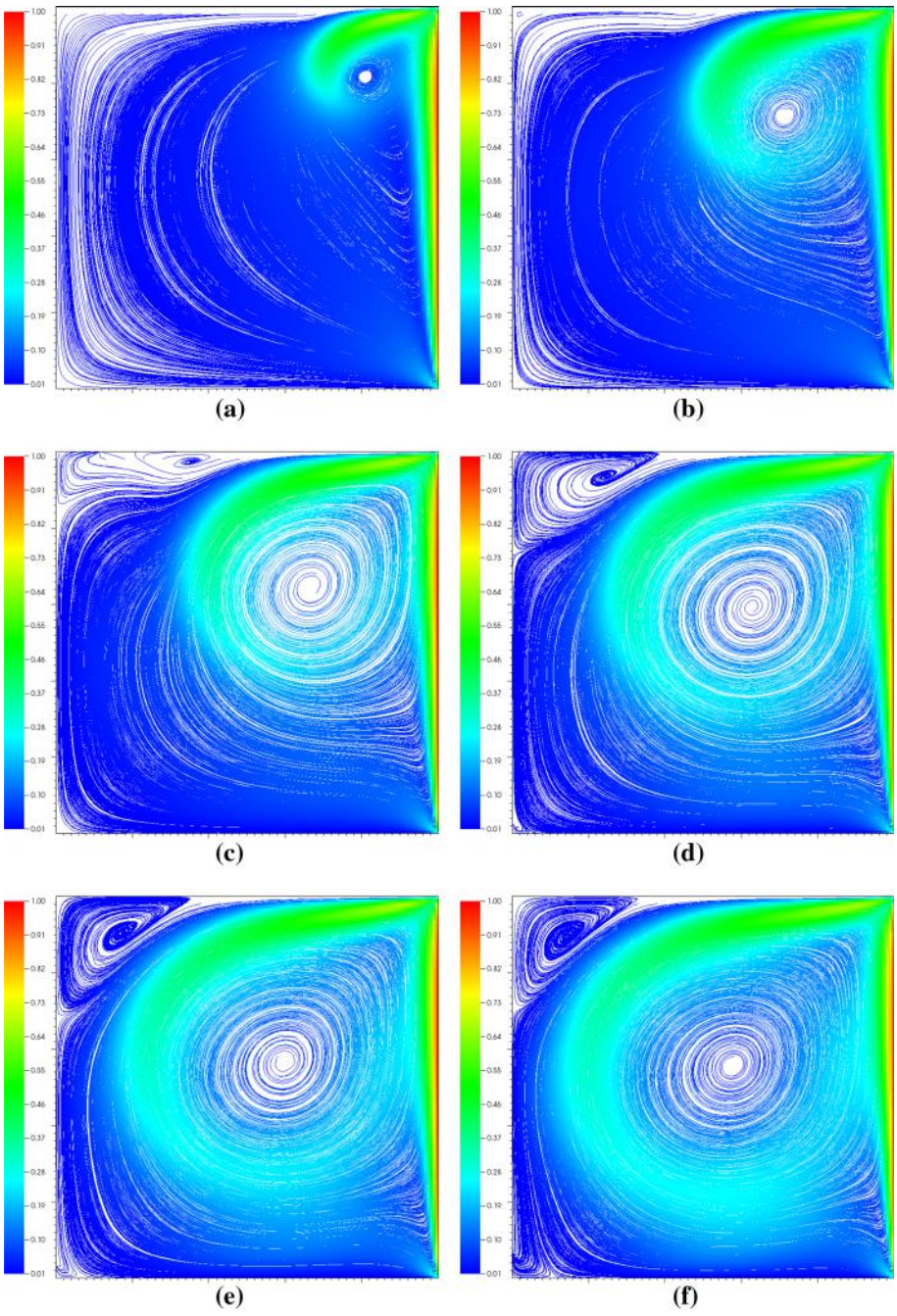

Fig. 6 Snapshots of streamlines for the 3D lid-driven cavity test inside the symmetry plane $z = 1$ on a uniform grid with $h = \frac{1}{128}$. *Different colors* represent different values of $\|\mathbf{u}\|_2$. **a** t = 2. **b** t = 4. **c** t = 6. **d** t = 8. **e** t = 10. **f** t = 12 (Color figure online)

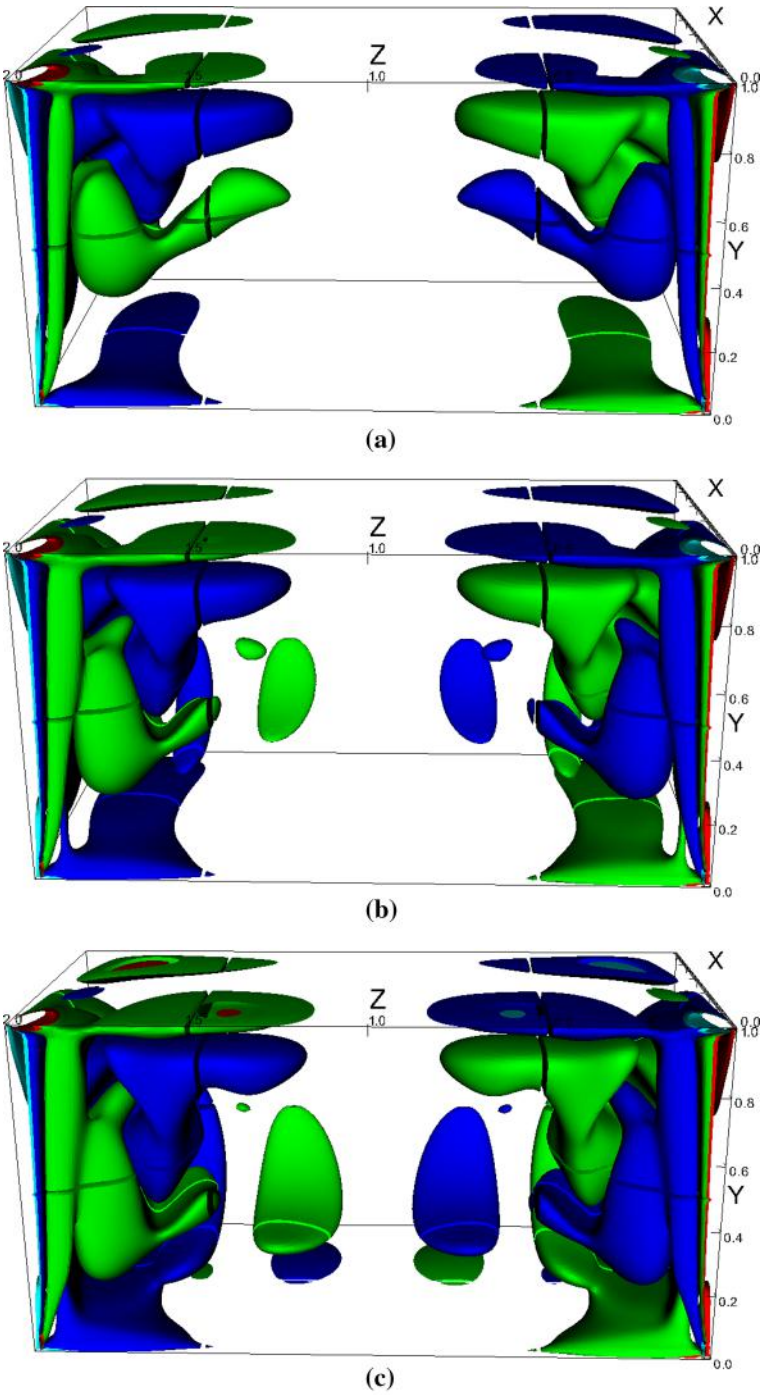

Fig. 7 Isosurfaces the x component of the vorticity vector for the 3D lid-driven cavity test on a uniform grid with $h = \frac{1}{128}$. The values of the vorticity component for the *red*, *orange*, *blue*, *cyan* surfaces are -0.50, -0.25, 0.25, and 0.50, respectively. **a** t = 8. **b** t = 10. **c** t = 12 (Color figure online)

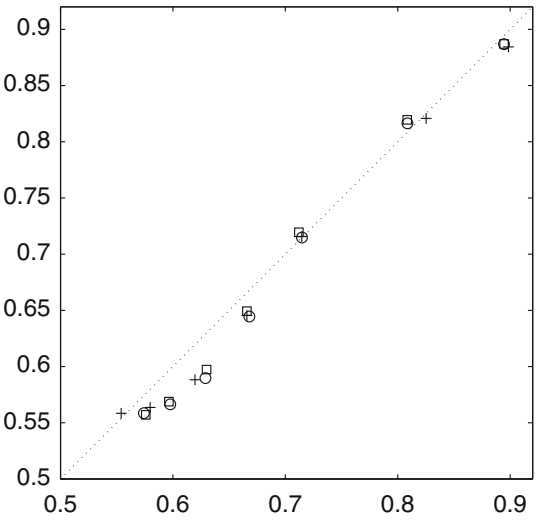

Fig. 8 A comparison of the center locations of the primary eddy in the symmetry plane $z = 0$ at time instances $t = 1, 2, 4, 6, 8, 10, 12$. The *circles* represent numerical results of this work, which are determined by minimizing the speed $\|\mathbf{u}\|_2$. The experimental and computational results of Guermond et. al. [19] are represented by the *crosses* and the *squares*, respectively

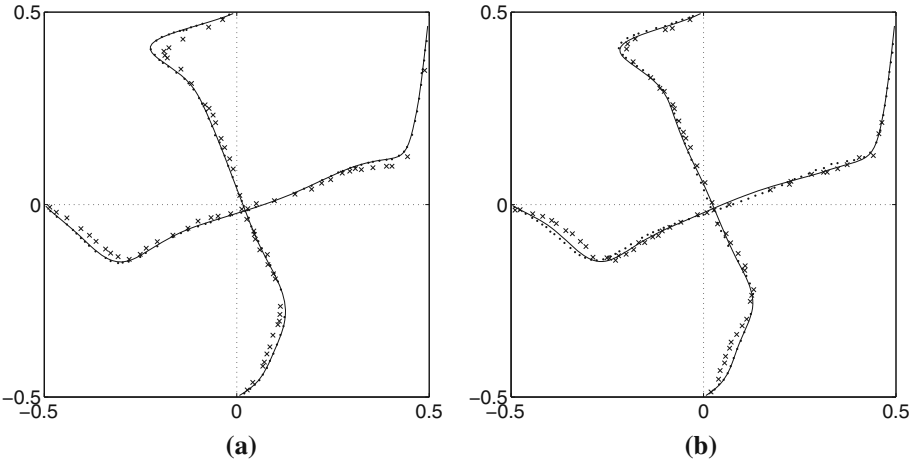

Fig. 9 A comparison of velocity profiles for the 3D lid-driven cavity test ($h = \frac{1}{128}$) at $t = 12$. *Solid lines* represent computational results of this work, i.e. $-\frac{1}{2}u$ as a function of $\frac{1}{2} - y$ and $\frac{1}{2}v$ as a function of $\frac{1}{2} - x$. The computational and experimental results of Guermond et. al. [19] are represented by the *solid dots* and *crosses*, respectively. **a** $z = 1$. **b** $z = 0.5$

7 An Efficiency Comparison

Within each time step, seventeen and eight linear systems have to be solved respectively for GePUP-IMEX and GePUP-ERK. This seemingly large computational expense can be justified by comparing the efficiency of GePUP-IMEX to that of a second-order method [35] using a simple formula derived in Sect. 7.1. Since GePUP-IMEX is about 1.5 times faster than GePUP-ERK, one can simply multiply speedup values of GePUP-IMEX by 1.5 over the second-order method to obtain those of GePUP-ERK.

Hereafter two assumptions are made:

(ECA-1) the time spent in solving the linear systems dominates that of explicitly evaluating discrete operators,
(ECA-2) the CPU time spent in solving a linear system is linearly proportional to the number of unknowns,

(ECA-1) is clearly reasonable for most numerical methods that solve the INSE. (ECA-2) holds if efficient multigrid algorithms are employed to solve the linear systems. Timing results in Table 8 also confirm their validity.

7.1 A Simple Formula for Comparing Methods with Different Convergence Rates

The *(relative) accuracy of a method* is defined as the relative difference between the computed solutions on two successively refined grids, or, if the exact solution is available, as the relative difference between computed solution and the exact solution. Given a desired accuracy ϵ, let $h_2(\epsilon)$ and $h_4(\epsilon)$ denote the grid size of a second-order method and that of the fourth-order method, respectively. Then

$$h_2 = \Theta\left(\epsilon^{\frac{1}{2}}\right), \quad h_4 = \Theta\left(\epsilon^{\frac{1}{4}}\right), \tag{88}$$

where $f = \Theta(g)$ means that $c_l g \le f \le c_u g$ for some constants $c_l, c_u > 0$.

For a given test, let T_2 and T_4 respectively denote the computational time of a second-order method and a fourth-order method to achieve a given accuracy ϵ. It follows from (ECA-1) and (ECA-2) that

$$T_2 = c_2\left(\frac{1}{h_2}\right)^{\mathrm{D}+1}, \quad T_4 = c_4\left(\frac{1}{h_4}\right)^{\mathrm{D}+1}, \tag{89}$$

where c_2 and c_4 are two positive constants, and the "+1" in the power coefficients of $\left(\frac{1}{h_2}\right)$ and $\left(\frac{1}{h_4}\right)$ comes from the fact that the initial condition has to be marched to the final solution over $\Theta(\frac{1}{h})$ time steps. Consequently, the *speedup* of a fourth-order method over a second-order method is

$$S_{4|2} := \frac{T_2}{T_4} = c_{4|2}\epsilon^{-\frac{\mathrm{D}+1}{4}}, \tag{90}$$

where the second identity follows from (88) and (89), and $c_{4|2}$ is a constant for the given test problem. Hence, the speedup as a function of the accuracy requirement is determined by the value of $c_{4|2}$, which can be deduced from the timing results of running the two methods with the same setup.

On the same grid, let E_2 and E_4 respectively denote the accuracy of the second-order method and the fourth-order method for running the same test problem; let n_T denote the ratio of the time spent by the fourth-order method to that by the second-order method. We assume that $E_4 \le E_2$, and both the order of accuracy and computing time have reached the asymptotic range. For the second-order method, let $n_{2\to4}$ denote the number of times that the grid has to be refined to improve its accuracy from E_2 to E_4, let r denote the refinement ratio of two adjacent grids. Then

$$n_{2\to4} = \log_{r^2}\frac{E_2}{E_4}, \tag{91a}$$

$$\frac{T_2}{T_4} = \frac{r^{(\mathrm{D}+1)n_{2\to4}}}{n_T}, \tag{91b}$$

where the power 2 in r^2 is the convergence rate of the second-order method, and the nominator in the RHS of (91b) follows from (89) and (91a). Recall T_2 and T_4 are the time needed by the two methods to reach the *same* accuracy; they are *not* the computing time corresponding to E_2 and E_4. It follows from (91b) and (90) that the values of $c_{4|2}$ for a fixed test problem can be calculated by

$$c_{4|2} = \frac{1}{n_T} (E_4)^{\frac{D+1}{4}} \left(\frac{E_2}{E_4} \right)^{\frac{D+1}{2}}, \tag{92}$$

where the given accuracy ϵ in (90) has been identified with E_4. In summary, the speedup of the fourth-order method to the second-order method as a function of the given accuracy requirement is

$$S_{4|2}(\epsilon) = \frac{1}{n_T} \left(\frac{E_2}{E_4} \right)^{\frac{D+1}{2}} \left(\frac{E_4}{\epsilon} \right)^{\frac{D+1}{4}}. \tag{93}$$

As the main utility of (93), the efficiency of fourth-order methods and second-order methods can be compared conveniently via the values of E_2, E_4, and n_T, all of which can be obtained by running the two methods on a single grid. The generalization of this formula to other convergence rates is straightforward and could be useful. As promised in the abstract and the introduction, the formula for the speedup of a ξth-order method over an ηth-order method in achieving the same accuracy ϵ is

$$S_{\xi|\eta}(\epsilon) = \frac{1}{n_T} \left(\frac{E_\eta}{E_\xi} \right)^{\frac{D+1}{\eta}} \left(\frac{E_\xi}{\epsilon} \right)^{\frac{(\xi-\eta)(D+1)}{\xi\eta}}, \tag{94}$$

where $\xi, \eta \in \mathbb{R}^+$, $\xi \geq \eta$, and $E_\xi \leq E_\eta$.

7.2 An Efficiency Comparison of GePUP-IMEX to a Second-Order Method

The second-order method chosen in this efficiency comparison is the one proposed by Martin, Colella, and Graves (MCG) [35], which is a further development of the classical second-order method by Bell, Colella, and Glaz (BCG) [6]. In both BCG and MCG, the convection term is calculated by a second-order Godunov method. As for temporal integration, MCG adopts the method by Twizell et. al. [53], which has a stronger stability than the Crank–Nicolson scheme used in BCG. MCG is implemented within the Chombo package [1], the source code of which is readily available [12]. Another reason for choosing MCG concerns efficiency. Linear systems in MCG are solved by efficient geometric multigrid methods similar to those discussed in Sect. 5.1. Also, the performance of the Chombo infrastructure has been very much optimized by cross-compilation: the low-level number-crunching subroutines are written in FORTRAN 77 and the upper-level software interface in C++. In summary, MCG and its Chombo implementation provide an excellent baseline of efficiency for many second-order methods.

After performing the numerical tests using the Chombo implementation of MCG, the author collects the L_2 errors and convergence rates of the velocity in Table 7, where the second-order accuracy is recovered in most tests. The time spent in performing these tests[12] are also recorded in Table 8, where the last three columns support the assumptions (ECA-1) and (ECA-2).

[12] The Chombo code is compiled with the switches "`OPT=HIGH DEBUG=FALSE MPI=TRUE`."

Table 7 The L_2 errors and convergence rates of the velocity for MCG [35]

	$\frac{1}{256}-\frac{1}{512}$	Rate	$\frac{1}{512}-\frac{1}{1024}$	Rate	$\frac{1}{1024}-\frac{1}{2048}$
single-vortex, Re$=2\times10^4$	3.36e−04	1.68	1.04e−04	2.02	2.57e−05
	$\frac{1}{128}-\frac{1}{256}$	Rate	$\frac{1}{256}-\frac{1}{512}$	Rate	$\frac{1}{512}-\frac{1}{1024}$
2D viscous-box, Re$=10^4$	9.41e−04	1.90	2.53e−04	1.98	6.41e−05
	$\frac{1}{64}-\frac{1}{128}$	Rate	$\frac{1}{128}-\frac{1}{256}$	Rate	$\frac{1}{256}-\frac{1}{512}$
2D viscous-box, Re$=100$	2.88e−04	1.99	7.28e−05	1.99	1.83e−05
	$\frac{1}{32}-\frac{1}{64}$	Rate	$\frac{1}{64}-\frac{1}{128}$	Rate	$\frac{1}{128}-\frac{1}{256}$
3D viscous-box, Re$=10^4$	3.07e−03	−0.26	3.68e−03	0.88	2.00e−03
3D viscous-box, Re$=100$	1.03e−03	1.94	2.69e−04	1.98	6.80e−05

The tests are exactly the same as those in Sects. 6.2 and 6.3. For the 2D viscous-box test on the intermediate grids, the error of MCG (7.28e−5) is about half of that (1.36e−4) of BCG [6]

Table 8 Time spent in performing the tests in Table 7 using MCG

	$\frac{1}{512}$	Ratio	$\frac{1}{1024}$	Ratio	$\frac{1}{2048}$
single-vortex, Re$=2\times10^4$	784	9.2	7207	8.6	61978
	$\frac{1}{256}$	Ratio	$\frac{1}{512}$	Ratio	$\frac{1}{1024}$
2D viscous-box, Re$=10^4$	33	7.8	259	8.1	2092
	$\frac{1}{128}$	Ratio	$\frac{1}{256}$	Ratio	$\frac{1}{512}$
2D viscous-box, Re$=100$	25	5.6	141	8.0	1128
	$\frac{1}{64}$	Ratio	$\frac{1}{128}$	Ratio	$\frac{1}{256}$
3D viscous-box, Re$=10^4$	70	18.8	1306	15.9	20747
3D viscous-box, Re$=100$	73	18.9	1380	16.2	22404

The integers are the CPU time in seconds running four parallel processes on the author's personal desktop with a single Intel® i7-3930k CPU. The 3D tests were also ran with four parallel processes since the CPU only has six physical cores. As for GePUP-IMEX, the time of performing the five tests on the finest grids is 239234, 8871, 5376, 96058, and 110452 in seconds, respectively. The corresponding values of n_T are thus 3.9, 4.2, 4.8, 4.6, and 4.9, respectively

For each row of Table 7, the error on the finest grid is denoted as E_2; the value of the corresponding E_4 is readily available from the tables in Sects. 6.2 and 6.3. As shown in Table 8, $n_T \in [3.9, 4.9]$ holds for all tests on the finest grids. This range of n_T is expected since there are 4 and 17 multigrid solves within each time step for MCG and GePUP-IMEX, respectively[13]. For simplicity, $n_T = 5$ is used throughout this work. With these numbers, $S_{4|2}$ can be calculated[14] by (93) for each given ϵ.

Table 9 shows values of the speedup of GePUP-IMEX over MCG for various tests. For relative accuracy as low as $\epsilon = 10^{-4}$, GePUP-IMEX is only moderately faster than MCG.

[13] This fact supports the methodology of efficiency comparison in [57], where the number of multigrid solves is used to measure the CPU time.

[14] As shown in Table 7, MCG has not reached its asymptotic convergence rates on the finest grid for the 3D viscous-box test with Re$=10^4$. Nonetheless, it is assumed that the convergence rates of MCG upon the very next grid refinement will reach 2. This assumption favors the performance of MCG and makes the values of the estimated speedup smaller. This footnote also applies to Tables 10, 11, and 12.

Table 9 The speedup $S_{4|2}$ of GePUP-IMEX over MCG [35] for achieving the same L_2 accuracy of the velocity

$S_{4\|2}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-7}$	$\epsilon = 10^{-8}$
single-vortex, Re $= 2 \times 10^4$	1.73e+00	9.71e+00	5.50e+01	3.07e+02	1.73e+03
2D viscous-box, Re $= 10^4$	5.03e+00	2.83e+01	1.59e+02	8.94e+03	5.03e+03
2D viscous-box, Re $= 100$	1.26e+01	7.11e+01	4.00e+02	2.25e+03	1.26e+04
3D viscous-box, Re $= 10^4$	8.65e+01	8.65e+02	8.65e+03	8.65e+04	8.65e+05
3D viscous-box, Re $= 100$	4.24e+01	4.24e+02	4.24e+03	4.24e+04	4.24e+05

The values of $c_{4|2}$ for these tests are 1.73e−03, 5.03e−03, 1.26e−02, 8.65e−03, and 4.24e−03, respectively

However, as ϵ gets smaller and smaller, the saving factor $S_{4|2}$ grows as a power function of $1/\epsilon$. In particular, $S_{4|2}$ is proportional to $1/\epsilon$ for the 3D tests.

All tests presented in this paper are performed on the author's personal desktop with an Intel® i7-3930 hexa-core CPU running at around 138 Giga-flops. The fastest computer in this world as of 2015/JUNE [37] is the Tianhe-2 computer in China with 3,120,000 cores and a maximum Linpack performance at 33, 862, 700 Giga-flops [37]. Clearly, an upper bound[15] of the speedup in switching from the author's personal desktop to Tianhe-2 is about 2.5×10^5. The fact that some numbers in Table 9 are greater than 2.5×10^5 leads to the following remark.

Remark 1 There exist test problems such that, to achieve certain accuracy requirements, the proposed fourth-order method running on the author's personal desktop is faster than the second-order method MCG [35] running on the fastest super-computer in the world!

Even for moderate accuracy requirements $\epsilon = 10^{-5}, 10^{-6}, 10^{-7}$, the benefit of the fourth-order method is still very appealing. Although the values of $S_{4|2}$ for other problems may differ from those in Table 9, the pattern of the power-function growth of $S_{4|2}$ stays the same. Even if accuracy is not of the primary concern, the fourth-order method is still preferred because it admits the use of much coarser grids to obtain the same accuracy.

In some cases, eight digits of accuracy for the velocity may be deemed as excessively deep. However, this "deep" accuracy of the velocity may be necessary to obtain a moderate accuracy of its derivatives such as the vorticity. In the results presented in Sects. 6.2 and 6.3, the error norms of the vorticity are much bigger than those of the velocity by two or three orders of magnitudes. The reason is that evaluating derivatives using the velocity of nearby control volumes cancels many digits of most significant bits. As h is reduced, the information of vorticity moves further and further to the right in the significand.[16] Hence, an accuracy considered deep for the velocity may not be deep at all for calculating the vorticity.

To quantify the arguments in the previous paragraph, the errors and convergence rates of the vorticity for MCG are shown in Table 10. Since the computed velocity is second-

[15] With the assumption of perfect scaling, this upper bound is very loose. The load balancing problem for high performance computing is notoriously difficult. For practical problems, it is often the case that the computing time can not be reduced any more once the number of processes reach tens of thousands, letting alone millions of them.

[16] When a derivative has to be calculated from the results of the primary variables, the order of accuracy of a method and the grid size for a specific problem needs to be balanced to avoid excessive lost of information from catastrophic cancellations. The hp-adaptivity in finite element method is a good example.

Table 10 The L_2 errors and convergence rates of the vorticity using MCG [35]

	$\frac{1}{256}-\frac{1}{512}$	Rate	$\frac{1}{512}-\frac{1}{1024}$	Rate	$\frac{1}{1024}-\frac{1}{2048}$
single-vortex, Re$=2\times10^4$	5.46e−02	1.41	2.06e−02	1.66	6.52e−03
	$\frac{1}{128}-\frac{1}{256}$	Rate	$\frac{1}{256}-\frac{1}{512}$	Rate	$\frac{1}{512}-\frac{1}{1024}$
2D viscous-box, Re$=10^4$	2.21e−01	1.13	1.01e−01	1.40	3.80e−02
	$\frac{1}{64}-\frac{1}{128}$	Rate	$\frac{1}{128}-\frac{1}{256}$	Rate	$\frac{1}{256}-\frac{1}{512}$
2D viscous-box, Re$=100$	1.65e−02	1.48	5.93e−03	1.49	2.11e−03
	$\frac{1}{32}-\frac{1}{64}$	Rate	$\frac{1}{64}-\frac{1}{128}$	Rate	$\frac{1}{128}-\frac{1}{256}$
3D viscous-box, Re$=10^4$	9.88e−02	−1.25	2.34e−01	−0.18	2.66e−01
3D viscous-box, Re$=100$	2.69e−02	1.38	1.03e−02	1.45	3.77e−03

The tests are the same as those in Sects. 6.3 and 6.2

Table 11 The speedup $S_{3.5|1.5}$ of GePUP-IMEX over MCG [35] for achieving the same L_2 accuracy of the vorticity

$S_{3.5\|1.5}$	$\epsilon=10^{-2}$	$\epsilon=10^{-3}$	$\epsilon=10^{-4}$	$\epsilon=10^{-5}$	$\epsilon=10^{-6}$
single-vortex, Re$=2\times10^4$	4.83e+00	6.72e+01	9.33e+02	1.30e+04	1.80e+05
2D viscous-box, Re$=10^4$	4.18e+01	5.81e+02	8.08e+03	1.12e+05	1.56e+06
3D viscous-box, Re$=10^4$	5.26e+02	1.76e+04	6.73e+05	2.25e+07	7.51e+08

The values of $c_{3.5|1.5}$ for these tests are 2.17e−2, 2.17e−1, and 5.40e−1, respectively

order accurate, the vorticity near the domain boundary is only first-order accurate. Overall, the L_2 convergence rates of the vorticity for MCG are at best 1.5, as confirmed by Table 10. A similar argument implies that the L_2 convergence rates of the vorticity computed by GePUP-IMEX is around 3.5. For the tests with high Reynolds number, the speedup of GePUP-IMEX over MCG for achieving the same L_2 accuracy of the vorticity is calculated by setting $\xi = 3.5, \eta = 1.5$ in (94). The values of $S_{3.5|1.5}$ are listed in Table 11 for the accuracy range of $\epsilon = [10^{-6}, 10^{-2}]$. Even for just two digits of accuracy of the vorticity in the 3D viscous-box test, GePUP-IMEX is at least 526 times faster than MCG! Furthermore, Table 11 strongly supports Remark 1 in that many numbers in it are greater than 2.5×10^5, the maximum speedup one could hope for in switching from a personal desktop to the fastest super-computer in the world. These comparison results are not surprising since *the speedup of a high-order method over a low-order method is a reciprocal power function of the expected accuracy whereas the expansion of computational resources via parallel computing is linear*.

Vortex dynamics is of great significance in turbulent flows, especially for regions near a no-slip boundary. Hence sometimes it is reasonable to use the max-norm as an accuracy measure of the vorticity. The speedup $S_{3|1}$ of GePUP-IMEX over MCG for achieving the same L_∞ accuracy of the vorticity is again calculated by (94) and its values are listed in Table 12. The numbers there provide a much stronger support to Remark 1 than Tables 9 and 11. The reader is invited to verify that the power coefficients of $1/\epsilon$ in the expression of $S_{3|1}$ is respectively 2 and $\frac{8}{3}$ for 2D and 3D, which explains the dramatic increase of $S_{3|1}$.

Table 12 The speedup $S_{3|1}$ of GePUP-IMEX over MCG [35] for achieving the same L_∞ accuracy of the vorticity

$S_{3\|1}$	$\epsilon = 10^{-2}$	$\epsilon = 10^{-3}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-6}$
single-vortex, $\mathrm{Re} = 2 \times 10^4$	1.89e+04	1.89e+06	1.89e+08	1.89e+10	1.89e+12
2D viscous-box, $\mathrm{Re} = 10^4$	2.30e+05	2.30e+07	2.30e+09	2.30e+11	2.30e+13
3D viscous-box, $\mathrm{Re} = 10^4$	3.66e+07	1.70e+10	7.88e+12	3.66e+15	1.70e+18

The values of $c_{3|1}$ for these tests are 1.89, 23.0, and 170.0, respectively

The discussion above is by no means a deny of the value of parallel computing. Instead, the author advocates that *truly fourth- and higher-order methods be preferably combined with parallel computing for a better efficiency in high performance computing*. Exactly how much computational time one could save in switching from a second-order method to a fourth-order method varies largely according to various factors such as

- does the application admit smooth solutions?
- which variables are of the primary concern?
- how many digits of accuracy the simulation is aiming for?

There probably exist real-world problems for which GePUP-IMEX would have no efficiency advantage over many second-order methods. Hence the author emphasizes that *the statement in Remark 1 is existential, not universal*.

Fourth- or higher-order accuracy might be extremely important for simulating flows where the physics depends largely on derivatives of the primary variables, e.g. curvature driven flows and boundary layer flows with high Reynolds number. For first-order methods based on finite volume or finite difference, the computed velocity converges, but the velocity gradient tensor does not; otherwise a second-order scheme can be easily deduced. Such a method would be inappropriate for simulating flows with high-Reynolds number, because the most important factor $\nabla \mathbf{u}$ that determines the physics of the boundary layer is computed with a constant error regardless of the grid size. Consequently, it is dubious whether this first-order method has converged to the *right* solution or not. Similar suspicions apply to second-order methods for curvature-driven flows.

As modern research of fluid mechanics tackles more and more complex problems with multiple length scales and time scales, the efficiency and accuracy demand on high performance computing will become higher and higher. In the author's opinion, fourth- and higher-order methods have a better chance to answer this challenge than lower-order methods. This is especially true when a high-order method is aided by parallel computing and adaptive mesh refinement.

8 Concluding Remarks

With the aid of a new concept called generic projections, a novel reformulation of the INSE is proposed by generalizing the unconstrained PPE formulation of Liu, Liu, and Pego [33]. As a prominent advantage of this GePUP formulation over previous second-order projection methods, a high-order time-integration algorithm can be treated as a "black-box" when coupled to high-order spatial discretization, and thus nonphysical auxiliary variables are not

needed in GePUP. In addition, the boundary conditions of the velocity and the pressure in the GePUP formulation have nothing to do with the coupled time integrator. To illustrate this advantage, two time-marching schemes, GePUP-ERK and GePUP-IMEX, are proposed for solving the INSE with no-slip conditions with fourth-order accuracy both in time and in space. For flows with low-Reynolds number, the GePUP-IMEX scheme treats the diffusion term implicitly in time to remove the restrictive constraints on the time-step size. For flows with high-Reynolds number, the explicit GePUP-ERK scheme achieves an even better efficiency. Unlike previous second-order projection methods, both GePUP-IMEX and GePUP-ERK can be generalized to even higher-order accuracy in a straightforward manner.

A simple formula (94) is proposed for efficiency comparison of methods with different convergence rates. Systematic arguments and timing results show that the efficiency of the proposed method could be vastly superior over that of a second-order method MCG [35]. In particular, as shown in Table 12, even if just two digits of accuracy are need for the vorticity near a no-slip boundary in a 3D viscous-box test with $\mathrm{Re} = 10^4$, the proposed fourth-order GePUP schemes running on a personal desktop are faster than the second-order method MCG running on the fastest supercomputer in this world!

A number of future research prospects follow. To further enhance the efficiency, the author is currently working on an adaptive version of the GePUP schemes, using the techniques presented in [57]. To achieve better accuracy and stability for convection-dominated problems, numerical calculation of the convection term could benefit from a WENO scheme [16,50], a remapping algorithm such as limiters [36,55], or an energy-conserving discretization [39,46]. Another useful extension of the GePUP schemes concerns other types of domains such as those with inlet and outlet boundaries. As long-term goals, the author will generalize the GePUP schemes to multiphase flows with irregular and moving boundaries, via incorporating highly accurate Lagrangian-flux algorithms [56] and the polygonal area mapping method [58,59,61].

Acknowledgments The author is grateful to two anonymous referees, whose comments lead to an improvement of the quality of this paper. The author also thanks Prof. Chi-Wang Shu for the extra time and effort he spent in handling the manuscript.

9 Appendix

Figure 10 shows the Matlab code that generates all the formulas for filling ghost cells and extrapolating cell averages to face averages on the domain boundary.

```
function [pg, derBndry] = extrapolate(order, bcType)
% Returns formulas for ghosts and domain face averages w.r.t interior cell averages.
% INPUT.   order: the interpolating order of the polynomial
%         bcType: 'None', 'Diri', or 'Neum'.
% OUTPUT.    pg : the expressions of the ghosts.
%       derBndry: derivatives of the intepolating polynomial at the boundary.
% RATIONALE. set the boundary at x=0, the polynomial to $f = \sum_ c_i x^i$,
%           then a Diri BC leads to $f = g+\sum_i c_i x^i$
%            and a Neum BC leads to $f = c_0+gx+\sum_i c_i x^i$.

syms f g h coef dfdn ii m mm mmm m4 m5;
if bcType=='None'
    ids = [1:order];   f = [ii; m; mm; mmm; m4; m5];
elseif bcType=='Diri'
    ids = [2:order];   f = [g; ii; m; mm; mmm; m4];
elseif bcType=='Neum'
    ids = [1,3:order]; f = [ii; dfdn*h; m; mm; mmm; m4];
else
    error('Invalid str for bc type!');
end
n = size(ids(:));

A = sym(eye(order,order));
for i=1:n
    lo = 0-i; hi = 1-i;  % the interior cells have boundaries at 0, -1, -2, ...
    for j=1:order        % integrate the polynomial over these cells.
        A(ids(i),j) = (hi^j-lo^j)/j;
    end
end
coef = A\f(1:order);
nGhosts = 2;
B = sym(zeros(nGhosts,floor(order))); % matrix to calculate ghosts expressions
for i=1:nGhosts
    lo = i-1; hi = i;    % the ghost cells have boundaries at 0, 1, and 2
    for j=1:order
        B(i,j) = (hi^j-lo^j)/j;
    end
end
pg = B*coef;
derBndry = coef;
denom = sym(ones(order,1));
% derBndry now has the coefs of the polynomial, multiply n! to the n-th derivative.
for i=2:order
    denom(i) = h^(i-1);
    derBndry(i:order) = (i-1)*derBndry(i:order);
end
derBndry = simplify(derBndry./denom);
```

Fig. 10 A Matlab code for generating the formulas in Sect. 3.3 for filling ghost cells and extrapolating cell averages to face averages on the domain boundary

References

1. Adams, M., Colella, P., Graves, D.T., Johnson, J., Keen, N., McCorquodale, T.J.L.D.F.M.P., Schwartz, D.M.P., Sternberg, T., Straalen, B.V.: Chombo Software Package for AMR Applications—Design Document. Technical report, Lawrence Berkeley National Laboratory. LBNL-6616E (2014)
2. Almgren, A., Aspden, A., Bell, J.B., Minion, M.L.: On the use of higher-order projection methods for incompressible turbulent flow. SIAM J. Sci. Comput. **35**(1), B25–B42 (2013)
3. Alonso-Mallo, I.: Runge–Kutta methods without order reduction for linear initial boundary value problems. Numer. Math. **91**, 577–603 (2002)

4. Ascher, U.M., Ruuth, S.J., Spiteri, R.J.: Implicit-explicit Runge–Kutta methods for time-dependent partial differential equations. Appl. Numer. Math. **25**, 151–167 (1997)
5. Baek, H., Karniadakis, G.E.: Sub-iteration leads to accuracy and stability enhancements of semi-implicit schemes for the Navier–Stokes equations. J. Comput. Phys. **230**(12), 4384–4402 (2011)
6. Bell, J.B., Colella, P., Glaz, H.M.: A second-order projection method for the incompressible Navier–Stokes equations. J. Comput. Phys. **85**, 257–283 (1989)
7. Bell, J.B., Colella, P., Howell, L.H.: An efficient second order projection method for viscous incompressible flow. In: AIAA 10th Computational Fluid Dynamics Conference, pp. 360–367 (1991)
8. Boscarino, S., Pareschi, L., Russo, G.: Implicit-explicit Runge–Kutta schemes for hyperbolic systems and kinetic equations in the diffusion limit. SIAM J. Sci. Comput. **35**(1), A22–A51 (2013)
9. Brandt, A.: Algebraic multigrid theory: the symmetric case. Appl. Math. Comput. **19**(1–4), 23–56 (1986). doi:10.1016/0096-3003(86)90095-0
10. Briggs, W.L., Henson, V.E., McCormick, S.F.: A Multigrid Tutorial. SIAM, 2nd edn. (2000). ISBN:0-89871-462-1
11. Brown, D.L., Cortez, R., Minion, M.L.: Accurate projection methods for the incompressible Navier–Stokes equations. J. Comput. Phys. **168**(2), 464–499 (2001)
12. Chombo.: Software for adaptive solutions of partial differential equations (April 2014). http://seesar.lbl.gov/ANAG/software.html. Version 3.2
13. Chorin, A.J.: Numerical solution of the Navier–Stokes equations. Math. Comput. **22**(104), 745–762 (1968)
14. Cozzi, E., Pego, R.L.: On optimal estimates for the Laplace–Leray commutator in planar domains with corners. Proc. Am. Math. Soc. **139**(5), 1691–1706 (2011)
15. E, W., Liu, J.-G.: Gauge method for viscous incompressible flows. Comm. Math. Sci. **1**(2), 317 (2003)
16. Gottlieb, S., Ketcheson, D.I., Shu, C.-W.: High order strong stability preserving time discretizations. J. Sci. Comput. **38**(3), 251–289 (2009)
17. Gresho, P.M., Sani, R.L.: On presssure boundary conditions for the incompressible Navier–Stokes equations. Int. J. Numer. Methods Fluids **7**, 1115–1145 (1987)
18. Griffith, B.E.: An accurate and efficient method for the incompressible Navier–Stokes equations using the projection method as a preconditioner. J. Comput. Phys. **228**(20), 7565–7595 (2009)
19. Guermond, J.L., Migeon, C., Pineau, G., Quartapelle, L.: Start-up flows in a three-dimensional rectangular driven cavity of aspect ratio 1:1:2 at Re = 1000. J. Fluid. Mech. **450**, 169–199 (2002)
20. Guermond, J.L., Minev, P., Shen, J.: An overview of projection methods for incompressible flows. Comput. Methods Appl. Mech. Eng. **195**(44–47), 6011–6045 (2006)
21. Gullbrand, J.: An Evaluation of a Conservative Fourth Order DNS Code in Turbulent Channel. Technical report, Center for Turbulence Research, Stanford University (2000)
22. Henshaw, W.D.: A fourth-order accurate method for the incompressible Navier–Stokes equations on overlapping grids. J. Comput. Phys. **113**, 13–25 (1994)
23. Hokpunna, A., Manhart, M.: Compact fourth-order finite volume method for numerical solutions. J. Comput. Phys. **229**, 7545–7570 (2010)
24. Johnston, H., Liu, J.-G.: Accurate, stable and efficient Navier–Stokes solvers based on explicit treatment of the pressure term. J. Comput. Phys. **199**(1), 221–259 (2004)
25. Karniadakis, G., Israeli, M., Orszag, S.: High-order splitting methods for the incompressible Navier–Stokes equations. J. Comput. Phys. **97**(2), 414–443 (1991)
26. Kassam, A.-K., Trefethen, L.: Fourth-order time-stepping for stiff PDEs. SIAM J. Sci. Comput. **26**(4), 1214–1233 (2005)
27. Kennedy, C.A., Carpenter, M.H.: Additive Runge–Kutta schemes for convection–diffusion–reaction equations. Appl. Numer. Math. **44**, 139–181 (2003)
28. Kim, J., Moin, P.: Application of a fractional-step method to incompressible Navier–Stokes equations. J. Comput. Phys. **59**(2), 308–323 (1985)
29. Kleiser, L., Schumann, U.: Treatment of incompressibility and boundary conditions in 3-D numerical spectral simulations of plane channel flows. In: Proceedings of the Third GAMM—Conference on Numerical Methods in Fluid Mechanics, vol **2** of Notes on Numerical Fluid Mechanics, pp. 165–173. Springer (1980). ISBN:978-3-528-08076-1
30. Knikker, R.: Study of a staggered fourth-order compact scheme for unsteady incompressible viscous flows. Int. J. Numer. Methods Fluids **59**(10), 1063–1092 (2009)
31. Leriche, E., Perchat, E., Labrosse, G., Deville, M.: Numerical evaluation of the accuracy and stability properties of high-order direct Stokes solvers with or without temporal splitting. J. Sci. Comput. **26**(1), 25–43 (2006)
32. LeVeque, R.J.: Finite Difference Methods for Ordinary Differential Equations. SIAM, Philadelphia (2007). ISBN 978-0-898716-29-0

33. Liu, J.-G., Liu, J., Pego, R.L.: Stability and convergence of efficient Navier–Stokes solvers via a commutator estimate. Commun. Pure Appl. Math. **60**, 1443–1487 (2007)
34. Liu, J.-G., Liu, J., Pego, R.L.: Stable and accurate pressure approximation for unsteady incompressible viscous flow. J. Comput. Phys. **229**(9), 3428–3453 (2010)
35. Martin, D.F., Colella, P., Graves, D.: A cell-centered adaptive projection method for the incompressible Navier–Stokes equations in three dimensions. J. Comput. Phys. **227**, 1863–1886 (2008)
36. McCorquodale, P., Colella, P.: A high-order finite-volume method for hyperbolic conservation laws on locally-refined grids. Commun. Appl. Math. Comput. Sci. **6**(1), 1–25 (2011)
37. Meuer, H., Strohmaier, E., Dongarra, J., Simon, H.: Top 500 supercomputer sites. (April 2012). http://www.top500.org
38. Minion, M.L.: Semi-implicit spectral deferred correction methods for ordinary differential equations. Commun. Math. Sci. **1**(3), 471–500 (2003)
39. Morinishi, Y., Lund, T.S., Vasilyev, O.V., Moin, P.: Fully conservative higher order finite difference schemes for incompressible flow. J. Comput. Phys. **143**(1), 90–124 (1998)
40. Ohlsson, J., Schlatter, P., Fischer, P., Henningson, D.: Direct numerical simulation of separated flow in a three-dimensional diffuser. J. Fluid Mech. **650**, 307–318 (2010)
41. Ohlsson, J., Schlatter, P., Fischer, P.F., Henningson, D.S.: Stabilization of the spectral-element method in turbulent flow simulations. In: Hesthaven, J.S., Ronquist, E.M. (eds.) Spectral and High Order Methods for Partial Differential Equations, volume **76** of Lecture Notes in Computational Science and Engineering, pp. 449–458. Springer, Berlin (2011)
42. Orszag, S.A., Israeli, M., Deville, M.O.: Boundary conditions for incompressible flows. J. Sci. Comput. **1**(1), 75–111 (1986)
43. Pereira, J.M.C., Kobayashi, M.H., Pereira, J.C.F.: A fourth-order-accurate finite volume compact method for the incompressible Navier-Stokes solutions. J. Comput. Phys. **167**(1), 217–243 (2001)
44. Portero, L., Jorge, J.C., Bujanda, B.: Avoiding order reduction of fractional step Runge–Kutta discretizations for linear time dependent coefficient parabolic problems. Appl. Numer. Math. **48**, 409–424 (2004)
45. Sanderse, B., Koren, B.: Accuracy analysis of explicit Runge–Kutta methods applied to the incompressible Navier–Stokes equations. J. Comput. Phys. **231**, 3041–3063 (2012)
46. Sanderse, B., Verstappen, R.W.C.P., Koren, B.: Boundary treatment for fourth-order staggered mesh discretizations of the incompressible Navier–Stokes equations. J. Comput. Phys. **257**, 1472–1505 (2014)
47. Shankar, P.N., Deshpande, M.D.: Fluid mechanics in the driven cavity. Annu. Rev. Fluid Mech. **32**, 93–136 (2000)
48. Shirokoff, D., Rosales, R.R.: An efficient method for the incompressible Navier–Stokes equations on irregular domains with no-slip boundary conditions, high order up to the boundary. J. Comput. Phys. **230**(23), 8619–8646 (2011)
49. Shishkina, O., Wagner, C.: A fourth order finite volume scheme for turbulent flow simulations in cylindrical domains. Comput. Fluids **36**(2), 484–497 (2007)
50. Shu, C.-W.: High order weighted essentially nonoscillatory schemes for convection dominated problems. SIAM Rev. **51**(1), 82–126 (2009)
51. Trebotich, D.P., Colella, P.: A projection method for incompressible viscous flow on moving quadrilateral grids. J. Comput. Phys. **166**, 191–217 (2001)
52. Trottenberg, U., Oosterlee, C., Schuller, A.: Multigrid. Elsevier Academic Press (2001). ISBN:0-12-701070-X
53. Twizell, E.H., Gumel, A.B., Arigu, M.A.: Second-order, L0-stable methods for the heat equation with time-dependent boundary conditions. Adv. Comput. Math. **6**, 333–352 (1996)
54. Verstappen, R., Veldman, A.: Direct numerical simulation of turbulence at lower costs. J. Eng. Math. **32**(2–3), 143–159 (1997)
55. Zerroukat, M., Wood, N., Staniforth, A.: The parabolic spline method (PSM) for conservative transport problems. Int. J. Numer. Meth. Fluids **51**, 1297–1318 (2006)
56. Zhang, Q.: Highly accurate Lagrangian flux calculation via algebraic quadratures on spline-approximated donating regions. Comput. Methods Appl. Mech. Engrg. **264**, 191–204 (2013)
57. Zhang, Q.: A fourth-order approximate projection method for the incompressible Navier–Stokes equations on locally-refined periodic domains. Appl. Numer. Math **77**(C), 16–30 (2014)
58. Zhang, Q., Fogelson, A.: Fourth-order interface tracking in two dimensions via an improved polygonal area mapping method. SIAM J. Sci. Comput. **36**, A2369–A2400 (2014)
59. Zhang, Q., Fogelson, A.: MARS: An analytic framework of interface tracking via mapping and adjusting regular semi-algebraic sets. SIAM J. Numer. Anal. (2016, to appear)
60. Zhang, Q., Johansen, H., Colella, P.: A fourth-order accurate finite-volume method with structured adaptive mesh refinement for solving the advection–diffusion equation. SIAM J. Sci. Comput. **34**(2), B179–B201 (2012)

61. Zhang, Q., Liu, P.L.-F.: A new interface tracking method: the polygonal area mapping method. J. Comput. Phys. **227**(8), 4063–4088 (2008)
62. Zhang, Q., Liu, P.L.-F.: Handling solid–fluid interfaces for viscous flows: explicit jump approximation vs. ghost cell approaches. J. Comput. Phys. **229**(11), 4225–4246 (2010)