Journal of Computational Physics 229 (2010) 3543–3572

Contents lists available at ScienceDirect

Journal of Computational Physics

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

High-order accurate solution of the incompressible Navier–Stokes equations on massively parallel computers

R. Henniger *, D. Obrist, L. Kleiser

Institute of Fluid Dynamics, ETH Zurich, 8092 Zürich, Switzerland

ARTICLE INFO

Article history:
Received 19 May 2009
Received in revised form 3 December 2009
Accepted 12 January 2010
Available online 22 January 2010

MSC:
65M06
76M20

Keywords:
Incompressible flow
Implicit time integration
Iterative solution
Preconditioner
Massively parallel
High-order
Finite differences
Direct numerical simulation
Transition
Turbulent channel flow

ABSTRACT

The emergence of "petascale" supercomputers requires us to develop today's simulation codes for (incompressible) flows by codes which are using numerical schemes and methods that are better able to exploit the offered computational power. In that spirit, we present a massively parallel high-order Navier–Stokes solver for large incompressible flow problems in three dimensions. The governing equations are discretized with finite differences in space and a semi-implicit time integration scheme. This discretization leads to a large linear system of equations which is solved with a cascade of iterative solvers. The iterative solver for the pressure uses a highly efficient commutation-based preconditioner which is robust with respect to grid stretching. The efficiency of the implementation is further enhanced by carefully setting the (adaptive) termination criteria for the different iterative solvers. The computational work is distributed to different processing units by a geometric data decomposition in all three dimensions. This decomposition scheme ensures a low communication overhead and excellent scaling capabilities. The discretization is thoroughly validated. First, we verify the convergence orders of the spatial and temporal discretizations for a forced channel flow. Second, we analyze the iterative solution technique by investigating the absolute accuracy of the implementation with respect to the different termination criteria. Third, Orr–Sommerfeld and Squire eigenmodes for plane Poiseuille flow are simulated and compared to analytical results. Fourth, the practical applicability of the implementation is tested for transitional and turbulent channel flow. The results are compared to solutions from a pseudospectral solver. Subsequently, the performance of the commutation-based preconditioner for the pressure iteration is demonstrated. Finally, the excellent parallel scalability of the proposed method is demonstrated with a weak and a strong scaling test on up to $\mathcal{O}(10^4)$ processing units and $\mathcal{O}(10^{11})$ grid points.

1. Introduction

Over the past few decades, the high-fidelity simulation of incompressible viscous time-dependent flows in three dimensions has been established as an important tool for studying fundamental phenomena in canonical flow configurations [22,32]. With the continued rapid growth in computational power, larger and larger simulations become possible (e.g. [19,46]). At the same time, the change from large shared-memory supercomputing systems to massively parallel distributed-memory platforms has prompted new programming paradigms which require the application of different

* Corresponding author. Tel.: +41 44 632 0396; fax: +41 44 632 1147.
E-mail address: henniger@ifd.mavt.ethz.ch (R. Henniger).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.01.015

numerical methods. "Petascale systems" (supercomputers which can perform over 10^{15} floating-point operations per second, Flop/s) will become more widely available in the next few years. To be able to use these computational platforms for high-fidelity simulations, we have to employ new numerical tools.

The simulation code presented in this work is an efficient solver for large incompressible flow problems. It uses high-order discretization schemes and is optimized specifically for massively parallel supercomputers. Our choices for the parallelization strategy, solution algorithms and discretization schemes result from the targeted flow problems and type of computing platforms. The computer architecture, problem structure (size, boundary conditions, etc.), discretization schemes and numerical methods for solving large systems of equations are tightly connected factors which determine the overall efficiency of a simulation code and cannot be considered separately.

At this point, it is important to define the term "efficient" more clearly. Generally, we assume that the computational effort in terms of floating-point operations for a given accuracy of the numerical solution is already minimized such that only the parallelization can alter the overall efficiency of a parallel simulation code. Hence, "efficient" could refer to the ability to solve a problem of given size as fast as possible, but it could also mean that the overall time to solve the problem remains (nearly) constant when the problem size and the number of processors are increased at the same rate. The former definition relates to the *strong scalability* of a parallel simulation code, whereas the latter requires *weak scalability*. The present work focuses on weak scaling properties. This goal is in accordance with the trend towards larger and larger numerical simulations carried out on computers with more and more processors.

Many canonical flow problems in simple geometries allow the application of the fast Fourier transform (FFT) for the numerical solution of the governing equations. The FFT is typically employed to solve Poisson problems (which are associated with the incompressibility constraint) in spectral space. This approach is straightforward for Fourier or Fourier–Chebychev pseudospectral methods (e.g. [28,31]) and it can also be used by fast Poisson solvers for non-spectral methods (e.g. [38]). Since the solution of the Poisson problems is typically the most expensive part of the numerical solution, the high efficiency of this approach yields very fast simulation codes. Other direct or iterative solution techniques, except multigrid, are usually not competitive with such Poisson solvers [13] which are well suited for large shared-memory supercomputers with vector processors. With the advent of ever larger massively parallel supercomputers which offer thousands of distributed processing units these methods have lost some of their appeal due to their limited scalability. We will discuss in Section 3 that especially global discretization schemes (such as the Fourier, Fourier–Chebychev pseudospectral method or compact finite differences) may impose some limits on the weak scalability of simulation codes. The same holds for local methods if fast Poisson solvers are employed. Therefore, we propose to use local discretization schemes in all spatial directions together with iterative, multigrid-based solvers [13] for the algebraic equations. This approach offers comparable levels of numerical accuracy as spectral methods, but may offer better scalability properties.

The governing equations are discretized in time with a semi-implicit time integration scheme in primitive variables. The resulting linear system of equations is solved in each (sub-)time step after forming the Schur complement problem [47] for the pressure. To solve it iteratively we employ a sophisticated preconditioner which has been used before for steady-state problems [11]. This procedure requires solutions of secondary Poisson and Helmholtz problems which are solved iteratively as well. We demonstrate that a careful choice of (adaptive) termination criteria for the different problems leads to a reduced computational effort. Overall, the simulation code consists of a collection of carefully selected and tuned numerical methods in the spirit of *best practices*.

The implementation presented in this paper is limited to incompressible flows in rectangular domains with a non-uniform Cartesian grid. These limitations have been chosen for simplicity and clarity and we would like to emphasize that the generalization to more complex geometries using the immersed interface method [26], the immersed boundary method [35] and/or curvilinear orthogonal coordinates [5] is relatively straightforward. The numerical framework could also be used to simulate more complex physics (e.g. miscible multiphase flow [15,16] or compressible flows).

This paper orients itself roughly on the work of Brüger et al. [5] whose work was similarly motivated (except for our focus on parallel efficiency). Other central differences to their work are: the choice of the preconditioner for the pressure problem, sharper (adaptive) termination criteria for the iterative solvers, and the application of high-order finite difference schemes in all three directions (which is a direct result of our focus on massively parallel computing platforms). For an overview on other high-order solvers for the Navier–Stokes equations we refer the reader to the introduction in Brüger et al. [5] and references therein.

The remainder of this paper is structured as follows: Section 2 defines the governing equations and boundary conditions. We discuss the parallelization and solution strategy in Section 3. Section 4 introduces the temporal and spatial discretization schemes. The resulting large system of linear equations is solved by a cascade of iterative solvers with preconditioners (Section 5). We validate our implementation in Section 6, test the performance of the preconditioner for the pressure problem in Section 7 and present weak and strong scaling tests in Section 8. Section 9 concludes this paper.

2. Governing equations

The present work describes a simulation code which solves the Navier–Stokes equations for incompressible flows written in non-dimensional form,

$$\frac{\partial u}{\partial t} + \underbrace{(u \cdot \nabla)u}_{\equiv -\mathcal{N}(u)} = -\nabla p + \underbrace{Re^{1}\Delta u}_{\equiv \mathcal{L}u}, \tag{1a}$$

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

for an initial condition $u(t=0) = u^{(0)}$ and appropriately chosen boundary conditions for u. The variable p stands for the pressure and $u = [u_1, u_2, u_3]$ for the velocity whose components are aligned with the Cartesian coordinate directions $[x_1, x_2, x_3]$. The Reynolds number is defined as $Re = UL/\nu$ with some reference velocity U, length scale L and the kinematic viscosity ν. The linear viscous terms in Eq. (1a) are named $\mathcal{L}u$, whereas all other terms (except for the time derivative and the pressure gradient) are collected in $\mathcal{N}(u)$. In matrix form the governing equations are written as

$$\frac{\partial}{\partial t}\begin{bmatrix} u \\ 0 \end{bmatrix} + \begin{bmatrix} -\mathcal{L} & \mathcal{G} \\ \mathcal{D} & 0 \end{bmatrix}\begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} \mathcal{N}(u) \\ 0 \end{bmatrix} \tag{2}$$

where $\mathcal{D}$ and $\mathcal{G}$ are the divergence and gradient operator, respectively. An equation for the pressure is obtained by applying the divergence operator to Eq. (1a),

$$\mathcal{D}\mathcal{G}p = \mathcal{D}\mathcal{N}(u). \tag{3}$$

The formal solution of this Poisson equation could be introduced directly into Eq. (1a) to eliminate the pressure. In that sense, the pressure can be seen as an auxiliary variable which is necessary to enforce incompressibility. The pressure Eq. (3) is inherently ill-posed, since the pressure appears only in the gradient operator $\mathcal{G}$, such that the absolute pressure level is never specified. However, the pressure gradient $\mathcal{G}p$ is unique and the Navier–Stokes equations with suitable velocity boundary conditions form a well-posed problem for the velocity u.

In the present implementation, we will solve a discretized form of Eq. (2). From this discretized system of equations, we will derive a discretized equation for the pressure such that no pressure boundary conditions need to be specified explicitly. The resulting pressure equation is different from a direct discretization of Eq. (3) due to the velocity and pressure boundary conditions and the discretization of $\partial/\partial t$.

The governing equations are solved in a domain Ω with the boundary $\partial\Omega$. Different types of velocity boundary conditions can be formulated on the condition that they lead to a well-posed problem for u. Such boundary conditions must be either of Dirichlet-, Neumann-, or Robin-type [43]. Neumann boundary conditions can lead to an ill-posed problem in certain situations [43]. A special spatial discretization is required in the vicinity of the boundaries, especially for high-order finite differences. In such cases, Dirichlet and Robin boundary conditions for the boundary-normal velocity component lead implicitly to Neumann or Neumann-like boundary conditions for the pressure. Neumann boundary conditions for that velocity component lead generally to an ill-posed problem since they allow a divergence-free solution on the boundary only if the other velocity boundary conditions are set consistently. Even then, pressure boundary conditions must be defined explicitly. This is obvious in the discrete case, since those rows of the pressure problem which correspond to the boundary points would be zero otherwise.

3. Solution technique, discretization and parallelization

Because the pressure Eq. (3) is elliptic, the discretization of Eq. (2) leads to a large system of equations which has to be solved in every (sub-)time step. This is typically the most time-consuming part of the simulation. If the problem gets too large to fit into the main memory of a single processor, the problem can be distributed to a larger number of processors[1] of a parallel computer which requires a *data decomposition* method to breakup the vectors and matrices into smaller blocks. The same needs to be done if the processor speed is too low to solve the problem within a given time.

The method for decomposing the computational domain is usually dictated by the choice of the spatial discretization scheme and by the solution technique. Hence, a complete *strategy* for solving the incompressible Navier–Stokes equations numerically consists of a data decomposition method, a discretization scheme and an appropriate solution technique for the resulting system of linear equations. We will compare such strategies in the following where we presume that the computational effort, i.e. the number of floating-point operations to solve the discrete equations at a given accuracy, is more or less identical for the different strategies if no parallelization is present.

3.1. Static and dynamic data decomposition

The classical *static* data decomposition scheme is sketched in Fig. 1 for the two-dimensional case. Each processor holds a contiguous sub-domain of the whole computational domain in its memory. These sub-domains are connected to their neighboring sub-domains by *ghost cells* which are located at the interface between two sub-domains. Each sub-domain contains certain portions of the global vectors and discrete operators (e.g. the diagonal blocks in a system of linear equations). The ghost cells correspond to the parts of the operator which cannot be distributed (e.g. the off-diagonal blocks in a system

[1] Throughout this paper, we use the term "processor" or "core" to describe a single processing unit and not a CPU or node with multiple cores.

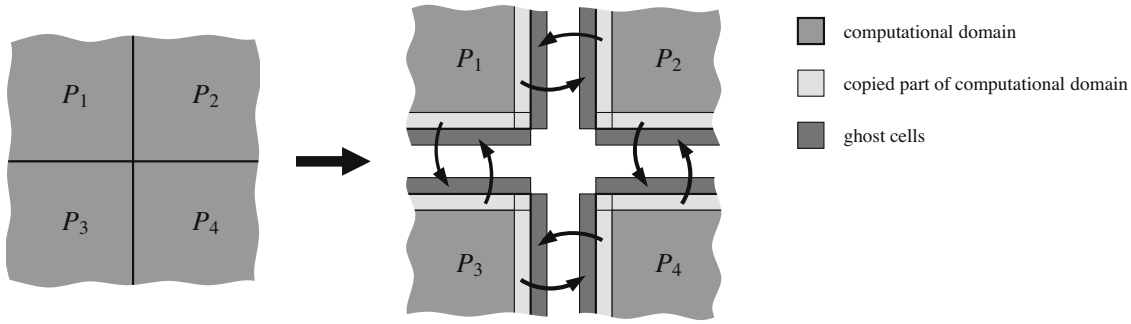

Fig. 1. Static data decomposition and ghost cell update between four processors.

Table 1
Complexities and scaling characteristics of the different parallelization approaches with the dimension d, the differentiation stencil width n, the number of grid points N^d and the number of processors P. Note that $d \log N = const. + \log P$ for $N^d \sim P$.

	Strategy A	Strategy B	+Multigrid	Strategy C
Computation	$\mathcal{O}(dnN^dP^{-1})$	$\mathcal{O}(dnN^dP^{-1}) + \mathcal{O}(dn^2N^{d-1}P^{3/d-1})$	$+\mathcal{O}(d \log P)$	$\mathcal{O}(dN^dP^{-1} \log N)$
Communication	Best:$\mathcal{O}(dnN^{d-1}P^{1/d-1})$ Worst: $\mathcal{O}(dnN^{d-1}P^{2/d-1})$	$\mathcal{O}(dnN^{d-1}P^{3/d-1})$	$+\mathcal{O}(d \log P) + \mathcal{O}(P^{1/d})$	$\mathcal{O}((d-1)N^dP^{1/d-1})$
Weak scaling $(d, n, N^d/P = const.)$	Best:$const.$ Worst: $const. + \mathcal{O}(P^{1/d})$	$const. + \mathcal{O}(P^{2/d})$	$+\mathcal{O}(\log P) + \mathcal{O}(P^{1/d})$	$const. + \mathcal{O}(\log P) + \mathcal{O}(P^{1/d})$
Strong scaling $(d, n, N^d = const.)$	Best:$\mathcal{O}(P^{-1}) + \mathcal{O}(P^{1/d-1})$ Worst: $\mathcal{O}(P^{-1}) + \mathcal{O}(P^{2/d-1})$	$\mathcal{O}(P^{-1}) + \mathcal{O}(P^{3/d-1})$	$+\mathcal{O}(\log P) + \mathcal{O}(P^{1/d})$	$\mathcal{O}(P^{-1}) + \mathcal{O}(P^{1/d-1})$

of linear equations). Before a global operator can be applied to a global vector, the data in the ghost cells has to be updated or synchronized with the corresponding data from the neighboring processors.

For non-local spatial discretization and/or solution methods (e.g. all Fourier methods) the static data decomposition is inefficient because these discretization schemes operate on a relatively large number of (or even all) grid points along a coordinate direction. Therefore, all data along this direction must belong to the same sub-domain. For operations in another direction the sub-domains have to be redefined. In that sense, this decomposition method is *dynamic*. It results in a repeated redistribution of the data. In two dimensions, this operation corresponds to a matrix transpose where the matrix entries are redistributed from a column-wise to a row-wise storage.

3.2. Discussion of different strategies

In principle, both data decomposition methods (static and dynamic) can be used for a massively parallel implementation. We will show in the following, however, that strategies based on a static decomposition may be better suited for the simulation of very large problems. This result applies for torus or mesh network topologies which are often used in modern supercomputer architectures (e.g. in Cray XT supercomputers with a 3D-torus). Other networks such as hypercube topologies [40] may lead to different results but will not be discussed here.

Ideally, we would like to increase the number of processors simultaneously with the problem size such that the simulation turn-around time remains constant (weak scaling). Strong scaling capabilities (smaller turn-around time for a constant problem size) is not the primary aim of our implementation (nevertheless, we will show in Section 8.2 that our strategy exhibits also excellent strong scaling capabilities). In order to assess the weak and strong scaling capabilities, we compare the numerical complexities of three different approaches listed in Table 1:

Strategy A A local discretization scheme (e.g. explicit finite differences, finite volumes, finite or spectral elements) with ghost cell updates using a static domain decomposition. Elliptic problems are solved with a multigrid-preconditioned Krylov subspace solver.

Strategy B Compact finite differences [25] parallelized by solving an auxiliary problem [30], i.e. a Schur complement problem, using a static domain decomposition. Elliptic problems are solved with a multigrid-preconditioned Krylov subspace solver.

Strategy C Fourier spectral discretization with FFTs and global data transpositions using a dynamic domain decomposition. Elliptic problems are solved in Fourier space.

In strategy A and B, we focus only on multigrid-preconditioned Krylov subspace solvers because they are very well suited for this kind of problem, see Section 5.2. We will first discuss the complexities of the differentiation operators of strategy A and B, followed by the complexity of multigrid and strategy C. For simplicity, we assume in this section that the computational domain with N^d grid points has the same dimension d as the network torus. The domain is distributed to P processors such that each sub-domain contains $N^d P^{-1}$ grid points. These sub-domains are cubes with edge length $NP^{-1/d}$ for the static decomposition and sticks of length N and width $NP^{-1/(d-1)}$ for the dynamic decomposition.

The computational complexity of strategy A is governed by the application of d differentiation stencils of length n on the $N^d P^{-1}$ data points in the sub-domain. Because only the ghost cells must be communicated to the nearest neighbours, the communication complexity is given for each dimension d by the product of the stencil width with the surface area of the sub-domain. However, this complexity assumes that the neighboring sub-domains are mapped to neighbouring processors (best processor mapping). In the worst case, communicating processor pairs have a distance of order $P^{1/d}$ ("Manhattan Distance" [29]) which will lead to increased traffic on the whole network. This network contention is characterized by the factor $P^{1/d}$. The processor mapping does not play the same role for the strategies B and C because these methods send data across the whole network anyway.

Compact finite differences [25] deserve a closer consideration because they offer higher accuracy compared to explicit finite differences while the computational effort increases only slightly. The implicit part of such schemes leads to narrowly banded matrices (provided that lexicographical ordering is used) which can be inverted efficiently with the Thomas algorithm, for instance. Such schemes are non-local and could be used in combination with a dynamic data decomposition method. However, it is more favorable to use a static data decomposition because we need to apply also multigrid which requires a static data decomposition as well as a large number of data exchanges. Strategy B is a popular method for using compact finite differences together with a static decomposition. Its computational complexity consists of the work for the banded sub-problems (first term) and the backward substitution of the (previously decomposed) Schur complement of size $nP^{1/d}$ which has to be done $dN^{d-1}P^{1/d-1}$ times. The communication complexity of strategy B is governed by $dN^{d-1}P^{1/d-1}$ transmissions of the solution vectors of the Schur problem of length $nP^{1/d}$. Network contention adds a factor $P^{1/d}$. As an alternative to strategy B, pipelining algorithms are available for compact finite differences (e.g. [9,36]). The algorithm described in [9] scales as well as explicit finite differences (strategy A) but is limited to periodic directions. The method proposed in [36] leads to load-balancing problems because of idling processors if the number of processors is not sufficiently small compared to the number of grid points (more precisely $P \ll N^{d-1}$). Therefore, pipelining algorithms will not be considered here.

To apply a differentiation scheme in the framework of a dynamic decomposition (strategy C) we need to transpose almost all data of size N^d over the network which has to be done $\mathcal{O}(d-1)$ times per sub-time step. The communication amount per processor is $\mathcal{O}(N^d P^{-1})$ which leads to the overall communication complexity $\mathcal{O}((d-1)N^d P^{1/d-1})$ on a torus network [40]. Note that the torus network always adds $P^{1/d}$ because ideal processor mapping is not possible.

So far, our analysis included only the differentiation operations as they are performed to evaluate the right-hand side of the pressure equation, for instance. For solving an elliptic equation, such as the pressure Eq. (3), additional computations and communications are required. For Fourier spectral methods (strategy C), this additional workload is negligible for the overall complexity compared to the FFTs because the solution of an elliptic equation with $N^d P^{-1}$ grid points in spectral space has complexity $\mathcal{O}(N^d P^{-1})$ (FFT: $\mathcal{O}(dN^d P^{-1} \log N)$). This is not the case for the strategies A and B. These strategies use iterative solvers such as multigrid. For multigrid, the complexity for computations and communications is governed by the work on the finest grids which scales like local discretization (strategy A). In parallel implementations, there is also the cost for the work on the coarse grids which have fewer grid points than processors. The computational work on the coarse grids has the complexity $\mathcal{O}(d \log P)$ because there are $\log P$ coarse levels and the work per level and processor has complexity $\mathcal{O}(d)$. Of course, the contribution of the coarse grids is usually very small. Similarly, the additional communication complexity on each of the $\log P$ coarse grids is $\mathcal{O}(d)$. Furthermore, the coarsest grids add the term $\mathcal{O}(P^{1/d})$ which accounts for the distance between the remaining communicating processors in the d-dimensional torus network. These results for multigrid are adapted from [29]. Apart from pure multigrid it is often beneficial to use Krylov subspace methods as primary solvers which may use multigrid as preconditioner (i.e. BiCGstab [42]). The costs for Krylov subspace methods with short recurrence such as BiCGstab consist of the costs for pure differentiation (strategy A or B), global vector–vector additions and scalar products which require communication across the whole network. In either case, these costs are already covered by the complexity of the differentiation plus multigrid and do not add any new dominant terms.

The weak and strong scaling properties of the different strategies follow immediately from the complexities and are also listed in Table 1. For the weak scaling we maintain d, n and the size of the sub-domains constant, $N^d/P = const.$, whereas we keep d, n and N fixed for the strong scaling. The absolute magnitudes of the individual terms depend on the used algorithms, the floating-point performance of the processors and the network bandwidth and latency. The best strong scaling capabilities are to be expected from strategy C. Strategy A can only compete with an optimal processor mapping. However, we find that strategy A without multigrid yields the best weak scaling properties, whereas strategy C is unable to yield a perfect weak scaling. Strategy B is neither competitive in the strong nor in the weak scaling.

The weak and strong scaling properties for multigrid appear to void the advantages of strategy A over strategy C, because strategy A with multigrid features now also algebraically growing terms. However, we have to stress again that the contributions by the coarse grids are very small compared to the massive data exchanges in a transposition. This will be illustrated in Section 8 where we will show that growing terms indeed do not contribute significantly to the overall cost which is mainly

dominated by the computations and the ghost cell update. Therefore, strategy A is our best candidate for good weak scalability. We use explicit finite differences in this study for simplicity reasons. This discretization allows a straightforward implementation of high-order schemes and offers some flexibility with respect to the choice of the geometry and boundary conditions. Other local discretization methods such as spectral elements would also be a viable choice since they offer similar properties, but will not be discussed in this work. Our solution approach for the discretized form of Eq. (1) is described in Section 5.2. It employs a combination of the previously mentioned iterative solvers and does not introduce any further complexities.

4. Discretization

4.1. Temporal discretization

The efficiency of a time integration scheme is determined by the computational cost for advancing the solution at a given level of accuracy by one time step and by the size of this time step. Typically, explicit schemes are less expensive per time step, but may require shorter time steps than implicit schemes for reasons of numerical stability. The maximum time step size for a stable time integration can be estimated from the Courant–Friedrichs–Lévy (CFL) condition. If only the convective terms $\mathcal{N}(u)$ in Eq. (1a) are taken into account, we set the convective time step limit for three spatial dimensions to

$$\Delta t \leqslant \frac{s_{\text{conv}}}{\max_\Omega\{|u_1|\hat{k}_{N,1}+|u_2|\hat{k}_{N,2}+|u_3|\hat{k}_{N,3}\}} \leqslant \text{CFL}_{\text{conv}} / \max_\Omega\left\{\frac{|u_1|}{\Delta x_1}+\frac{|u_2|}{\Delta x_2}+\frac{|u_3|}{\Delta x_3}\right\}. \tag{4}$$

If only the diffusive terms $\mathcal{L}u$ are considered, a viscous time step limit applies,

$$\Delta t \leqslant \frac{s_{\text{visc}}}{\max_\Omega\{(\hat{k}_{L,1}^2+\hat{k}_{L,2}^2+\hat{k}_{L,3}^2)/Re\}} \leqslant \text{CFL}_{\text{visc}} / \max_\Omega\left\{\frac{1}{Re}\left(\frac{1}{\Delta x_1^2}+\frac{1}{\Delta x_2^2}+\frac{1}{\Delta x_3^2}\right)\right\}. \tag{5}$$

In this paper, the CFL numbers are defined as

$$\text{CFL}_{\text{conv}} := \frac{s_{\text{conv}}}{\max_\Omega\{\hat{k}_{N,1}\Delta x_1+\hat{k}_{N,2}\Delta x_2+\hat{k}_{N,3}\Delta x_3\}} \quad \text{and} \quad \text{CFL}_{\text{visc}} := \frac{s_{\text{visc}}}{\max_\Omega\{(\hat{k}_{L,1}\Delta x_1)^2+(\hat{k}_{L,2}\Delta x_2)^2+(\hat{k}_{L,3}\Delta x_3)^2\}}. \tag{6}$$

The parameters s_{conv} and s_{visc} are the stability limits of the time integration scheme for oscillating and non-oscillating solutions, respectively, and $\hat{k}_{N,i}$ and $\hat{k}_{L,i}$ are the maximum modified wavenumbers of the spatial discretization of $\mathcal{N}(\cdot)$ and $\mathcal{L}$, respectively. The time step limits show that there is always a Reynolds number (and an according fine-grid spacing Δx) below which the viscous time step limit is more restrictive than the convective limit. In practice, this occurs at (locally) very fine spatial resolutions as we find them in direct numerical simulations. In large-eddy simulations the sub-grid scale model could theoretically introduce an even stronger limitation than the viscous time step limit although this is normally not the case.

Such time step restrictions can be avoided with an implicit time integration scheme. For $\mathcal{L}u$ this results in a linear system of equations, but for $\mathcal{N}(u)$ the implicit problem is nonlinear. Nevertheless, these fully implicit methods can be faster overall, as shown by Choi and Moin [7] for turbulent channel flow. Normally, this is not the case, in particular, when accuracy requirements impose a stronger limitation on the time step size than the stability limits (e.g. in transitional flows). In that case, the increased computational effort per time step cannot be compensated by a larger step size. Here, we use a semi-implicit scheme, where the nonlinear term $\mathcal{N}(u)$ is extrapolated in time (explicit time integration) while the linear part $\mathcal{L}u$ is treated implicitly, such that we obtain a linear system of equations. The time step Δt is then mainly constrained by the convective term $\mathcal{N}(u)$. In either case, the continuity Eq. (1b) is independent of time and must be satisfied at each time step which requires an implicit treatment. The same holds for the pressure gradient ∇p since Eqs. (1a) and (1b) are coupled through the pressure p. Such semi-implicit time integration schemes are analyzed in [21,41].

To obtain a temporal accuracy of at least second-order and to avoid the restrictive viscous time step limit, we use the absolutely stable Crank–Nicolson scheme (CN) for $\mathcal{L}u$. The explicit time integration of $\mathcal{N}(u)$ is done with a low-storage third-order accurate Runge–Kutta scheme (RK3-O3) by Wray [45]. The combination of the RK3-O3 scheme with the Crank–Nicolson scheme was reviewed and applied by Spalart et al. [39]. With the definitions $u^{(0)}=u(t)$, $u^{(3)}=u(t+\Delta t)$ and the intermediate solutions $u^{(1)}=u(t+a_1\Delta t)$, $u^{(2)}=u(t+(a_1+a_2+b_2)\Delta t)$, the semi-implicit low-storage CN-RK3 scheme reads

$$u^{(m)}-u^{(m-1)}+c_m\Delta t\mathcal{G}p^{(m)}=\frac{c_m}{2}\Delta t\left[\mathcal{L}u^{(m)}+\mathcal{L}u^{(m-1)}\right]+a_m\Delta t\mathcal{N}(u^{(m-1)})+b_m\Delta t\mathcal{N}(u^{(m-2)}),\quad m=1,2,3. \tag{7}$$

The coefficients a_m, b_m and c_m are listed in Table 2. For consistency with the Crank–Nicolson scheme, we must have $c_m=a_m+b_m$. This semi-implicit time integration leads to a coupled system of linear equations for the velocity $u^{(m)}$ and the pressure $p^{(m)}$ of the new sub-time level m,

$$\begin{bmatrix}\mathcal{H}^{(m)} & c_m\Delta t\mathcal{G}\\ \mathcal{D} & 0\end{bmatrix}\begin{bmatrix}u^{(m)}\\ p^{(m)}\end{bmatrix}=\begin{bmatrix}f(u^{(m-1)},u^{(m-2)})\\ 0\end{bmatrix},\quad m=1,2,3, \tag{8}$$

Table 2
Coefficients of the semi-implicit CN-RK3 time integration scheme.

m	a_m	b_m	c_m
1	8/15	0	8/15
2	5/12	−17/60	2/15
3	3/4	−5/12	1/3

where $\mathcal{H}^{(m)}$ is the Helmholtz operator,

$$\mathcal{H}^{(m)} = 1 - \frac{c_m}{2}\Delta t\mathcal{L}, \tag{9}$$

and $f(u^{(m-1)}, u^{(m-2)})$ contains the remainder of Eq. (7). The solution of the linear system (8) is typically by far the most time-consuming part of a simulation. For a purely explicit time integration, the Helmholtz operator becomes the identity operator (except for the boundary conditions) and the linear system (8) can be reduced to a single Poisson problem for the pressure.

4.2. Spatial discretization

For simplicity, the present work introduces a spatial discretization in a rectangular domain on a Cartesian grid with arbitrary grid stretching. Furthermore, a more complex domain can be realized easily using the immersed interface method [26] or the immersed boundary method [35] at no significant additional cost.

We use explicit finite differences of high convergence order for the spatial discretization of Eq. (8). This leads to a linear system of equations of the form

$$\begin{bmatrix} \mathbf{H} & \mathbf{G} \\ \mathbf{D} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix} \tag{10}$$

which has to be solved in each sub-time step of the time integration scheme (for the ease of writing, we have dropped the index m for the sub-time step level). The vector $\mathbf{u} = [\mathbf{u}_i]$ denotes the discretized velocity and $\mathbf{p}$ represents the discretized pressure. The matrices $\mathbf{D}$ and $\mathbf{G}$ are the discrete forms of the divergence operator $\mathcal{D}$ and the gradient operator $c_m\Delta t\mathcal{G}$, respectively[2]. The discrete Helmholtz operator is given by

$$\mathbf{H}^{(m)} = \mathbf{J} - \frac{1}{2}c_m\Delta t\mathbf{L} \tag{11}$$

where $\mathbf{L}$ stands for the discrete form of the linear operator $\mathcal{L}$. The matrix $\mathbf{J}$ is equal to the identity matrix $\mathbf{I}$ except that the rows which correspond to boundary points hold the velocity boundary conditions. The corresponding rows in $\mathbf{L}$ (as well as in $\mathbf{G}$) are left blank. Therefore, the pressure gradient $\mathbf{Gp}$ is only applied to the grid points of the inner field. In contrast, mass conservation is required everywhere and $\mathbf{Du} = \mathbf{0}$ is imposed at all points.

4.2.1. Staggered grid

As we will describe in Section 5.2, we derive an equation for the pressure by forming the Schur complement [47] of Eq. (10),

$$\mathbf{DH}^{-1}\mathbf{Gp} = \mathbf{DH}^{-1}\mathbf{f}. \tag{12}$$

If $\mathbf{DH}^{-1}\mathbf{G}$ has only one zero eigenvalue (and a rank deficit of one, respectively) which accounts for the undefined pressure constant then $\mathbf{DH}^{-1}\mathbf{G}$ is *h-elliptic* [2,4]. This property is not only necessary for a unique pressure (apart from the undefined constant) but is also a precondition for the convergence of multigrid-based solvers applied to such problems (cf. Section 5.2.2). A necessary condition for h-ellipticity is that the matrices $\mathbf{D}$ and $\mathbf{G}$ have at least the same rank as $\mathbf{DH}^{-1}\mathbf{G}$ (the Helmholtz matrices $\mathbf{H}$ are required to be non-singular anyway). Since $\mathbf{G}$ must additionally provide a right null space of rank one for the pressure constant, $\mathbf{G}$ must have exactly the same rank as $\mathbf{DH}^{-1}\mathbf{G}$ for h-ellipticity.

To judge whether a spatial discretization satisfies the requirements on $\mathbf{G} = [\mathbf{G}_1^T\mathbf{G}_2^T\mathbf{G}_3^T]^T$ it is convenient to map $\mathbf{G}_i$ first onto a uniform grid and to investigate the discrete Fourier transform (referred to as "symbol") of each matrix row of $\mathbf{G}_i' = \mathbf{M}_i\mathbf{G}_i$ with wavenumbers k_i subsequently. The rank of the right nullspace of $\mathbf{G}$ increases by one with each resolvable wavenumber $k = [k_1, k_2, k_3]$ for which all symbols of all $\mathbf{G}' = [\mathbf{G}'T_1\mathbf{G}_2'^T\mathbf{G}_3'^T]^T$ are zero. Hence, we can ensure the desired rank of $\mathbf{G}$ and $\mathbf{G}_i$ if each of the symbols of $\mathbf{G}_i'$ is non-zero for any resolvable *non-zero* wavenumber k_i. Similarly, it can be shown that $\mathbf{D}$ and its sub-matrices $\mathbf{D}_i$ have their full ranks if the discretization satisfies the same requirements as for $\mathbf{G}_i$.

On equidistant colocated grids (function values and their derivatives are stored on the same grid points) any odd spatial derivative of the grid cut-off wave number cannot be represented because information about either the amplitude or the phase shift of the wave is missing. Hence, the rank of the right null space of $\mathbf{G}$ becomes larger than one, such that $\mathbf{DH}^{-1}\mathbf{G}$

[2] In practice, it is more convenient to work directly with $\mathcal{G}$ and to substitute p with $c_m\Delta tp$ instead such that $\mathbf{G}$ (and all other operators building on it) needs to be discretized and stored only once.

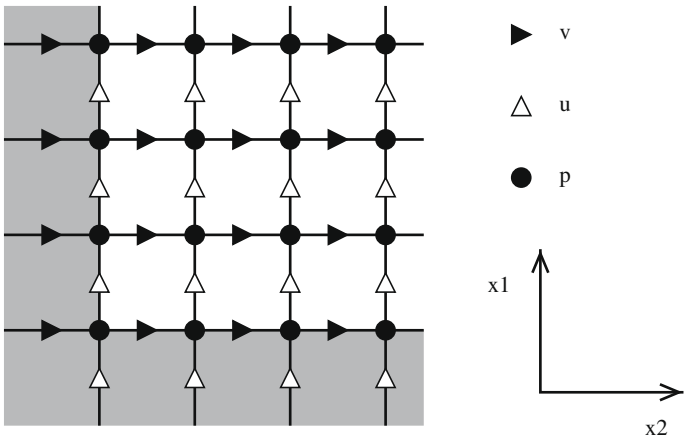

Fig. 2. Staggered grid in two dimensions near boundaries.

cannot be h-elliptic without appropriate measures. Fourier pseudospectral methods, for instance, allow an explicit handling of this mode by setting it simply to zero in spectral space [3]. Other discretizations require artificial "damping" of the higher wavenumbers to make $\mathbf{D}\mathbf{H}^{-1}\mathbf{G}$ h-elliptic [2]. On staggered grids, however, the symbols of **D** and **G** are typically non-zero for any resolvable non-zero wavenumber, such that $\mathbf{D}\mathbf{H}^{-1}\mathbf{G}$ normally is h-elliptic.

Therefore, we use finite differences on staggered grids for the velocity and the pressure (Fig. 2). We work with four subgrids: one for each velocity component and one for the pressure. The pressure grid is labeled 0 and the velocity grids are labeled 1, 2 or 3 (corresponding to the direction of the velocity component). The momentum equations are solved on the respective velocity grids, and the continuity equation is satisfied on the pressure grid. Hence, the discrete divergence operator **D** computes first derivatives on the grid 0 from values on the grids 1, 2 and 3, whereas the discrete gradient operator **G** computes first derivatives on the grids 1, 2 and 3 from values on grid 0. Dirichlet boundary conditions for the tangential velocity components can be applied directly to the grid points on the wall. The normal velocity component has to be imposed by interpolating between virtual grid points beneath the boundary and grid points just above the boundary.

The discrete form of the convective operator in $\mathcal{N}(u)$ involves products between velocity components and the first derivative of other velocity components. In this context, the first derivative on the grid i in the direction j is represented by the discrete operator $\mathbf{C}_{i,j}$,

$$\frac{\partial(\cdot)_i}{\partial x_j} \approx \mathbf{C}_{i,j}. \tag{13}$$

In general, the convective operator must transfer data between the velocity grids. To this end, the discrete interpolation operators $\mathbf{T}_{i,0}$ and $\mathbf{T}_{0,j}$ are introduced which interpolate function values from the pressure grid 0 onto the velocity grid i and values from the grid j onto grid 0, respectively. Hence, the local velocity on grid i is

$$\mathbf{u}_{j,i} = \mathbf{T}_{i,0}\mathbf{T}_{0,j}\mathbf{u}_j. \tag{14}$$

The components of $\mathbf{u}_{j,i}$ are multiplied with the components of the first derivative $\mathbf{C}_{i,j}\mathbf{u}_i$ and the discretized nonlinear terms $\mathcal{N}(u)$ in convective formulation take the final form

$$u_j \frac{\partial u_i}{\partial x_j} \approx \mathrm{diag}\{\mathbf{u}_{j,i}\}\mathbf{C}_{i,j}\mathbf{u}_i, \tag{15}$$

where $\mathrm{diag}\{\mathbf{u}_{j,i}\}$ is a diagonal matrix with the components of $\mathbf{u}_{j,i}$ as diagonal entries.

4.2.2. Finite difference stencils and non-uniform grids

Rather than applying metric terms to differentiation operators for equidistant grids, we introduce the grid stretching by computing the finite difference coefficients directly on the stretched grid for reasons of accuracy. To find the stencil coefficients with n points for the dth derivative at x^* we define the square matrix **B**,

$$\mathbf{B} = [b_{ij}] = [\Delta x_i^{j-1}], \quad i,j = 1,2,3,\ldots,n, \tag{16}$$

where $\Delta x_i = x^* - x_i$ (Fig. 3). The stencil coefficients c_i are then given by the $(1+d)$th column of the inverse transpose of **B**,

$$c_i = d![\mathbf{B}^{-T}]_{i,(1+d)}. \tag{17}$$

This scheme preserves the full accuracy for polynomials of order $n-1$. For $d=0$ we obtain an interpolation operator.

We use upwind-biased finite differences for the discretization of $\mathbf{C}_{i,j}$ to provide an anti-aliasing filter for under-resolved flows [27]. In these schemes, the outermost coefficient on the downwind side of the stencil is set to zero (Fig. 4). The

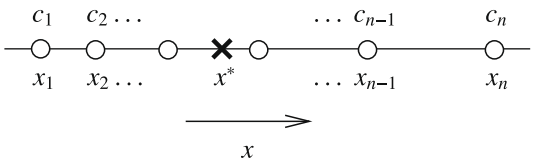

Fig. 3. Finite difference stencil with coefficients c_i and coordinates x_i. The derivative is computed at x^*.

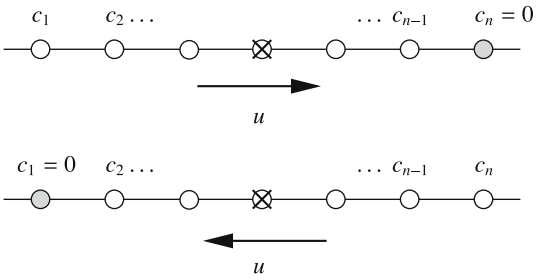

Fig. 4. Upwind-biased finite difference stencils.

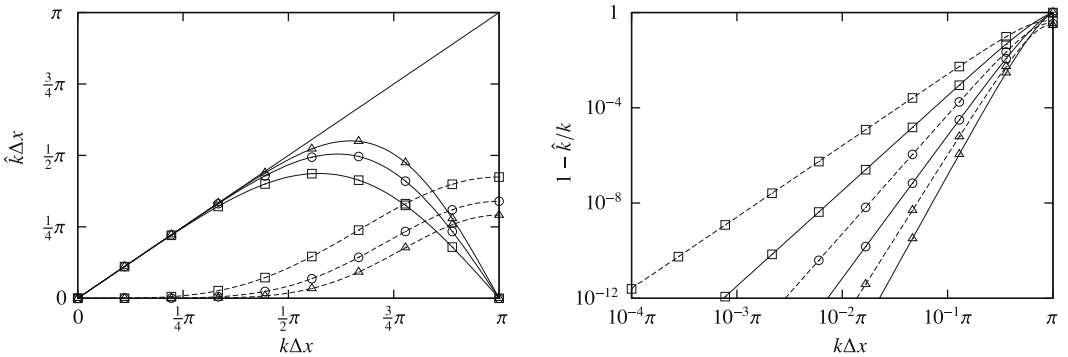

Fig. 5. Modified wave numbers of different upwind-biased schemes ((□) $\mathcal{O}(\Delta x^3)$, (○) $\mathcal{O}(\Delta x^5)$, (△) $\mathcal{O}(\Delta x^7)$; (————) real part, (- - -) imaginary part). Left: transfer function, right: relative error.

downwind side for the derivative $\mathbf{C}_{i,j}$ in the direction j on the grid i is indicated by the sign of the local convection velocity $\mathbf{u}_{j,i}$. The modified wave numbers of such schemes have an imaginary part which damps the solution especially at high wave numbers (Fig. 5). The real part of the modified wave number is exactly the same as for the central scheme with the same stencil width [27]. Therefore, upwinding does not affect the dispersion properties, but controls the accumulation of kinetic energy in the large wave numbers.

For the present implementation, we focus on achieving a certain overall *convergence order* rather than a certain overall *absolute discretization error* as done by Simens et al. [38], for instance. In that sense, the convergence order of all spatial operators should be more or less the same on each grid point to avoid extra work. Hence, we choose the same (central) stencil width n for all operators. In the interior of the domain, the following rules apply: If a variable and its derivative are defined on the same grid, which is the case for the operators $\mathbf{L}$, $\mathbf{H}$ and $\mathbf{C}$, the convergence order is $n-1$ for a central stencil (typically, n is an odd number). Only the upwind-biased schemes in $\mathbf{C}_{i,j}$ have a zero coefficient on the downwind side which gives a convergence order of $n-2$. All other operators ($\mathbf{D}$, $\mathbf{G}$ and $\mathbf{T}$) transfer information between different grids. In such cases, the stencil widths and the convergence orders are identical. We choose them to be $n-1$ in order to be consistent with the convergence orders of the other operators. In practice, we use five different sets of finite difference stencils (Table 3). As an example, the d3 differentiation scheme is sketched in Fig. 6.

5. Iterative solution

In this section, we discuss the solution of the system (10) by iterative methods. Usually, direct solvers (except FFT-based methods, cf. Section 3.2) have an unfavorable numerical complexity which limits them to problems with only small numbers of unknowns. However, there exist efficient iterative solvers which are much better suited for this type of problem. Furthermore, iterative methods allow a direct control of the solution accuracy.

Table 3
Convergence order (and number of non-zero coefficients) of the finite difference stencils on the first few grid points starting from the boundary. The first pair of numbers corresponds to the grid point on the boundary (colocated) or next to the boundary (staggered), cf. Fig. 6.

Name	Truncation error	Grid	Convergence order (number of coefficients)							
d1	$\mathcal{O}(\Delta x^1)$	Colocated	1(2)	1(3)	1(3)	...				
		Staggered	2(2)	2(2)	...					
d2	$\mathcal{O}(\Delta x^2)$	Colocated	2(3)	2(4)	3(5)	3(5)	...			
		Staggered	2(3)	4(4)	4(4)	...				
d3	$\mathcal{O}(\Delta x^3)$	Colocated	3(4)	3(5)	3(5)	5(7)	5(7)	...		
		Staggered	3(4)	4(4)	6(6)	6(6)	...			
d4	$\mathcal{O}(\Delta x^4)$	Colocated	4(5)	4(6)	4(6)	5(7)	7(9)	7(9)	...	
		Staggered	4(5)	4(4)	6(6)	8(8)	8(8)	...		
d5	$\mathcal{O}(\Delta x^5)$	Colocated	5(6)	5(7)	5(7)	5(7)	7(9)	9(11)	9(11)	...
		Staggered	5(6)	5(6)	6(6)	8(8)	10(10)	10(10)	...	

5.1. Schur complement formulation

Rather than applying an iterative solver to the original system (10) it is beneficial to transform or rearrange the problem (e.g. [23,24,44]) before applying preconditioned iterative solvers to the resulting (sub-)problems. To this end, we eliminate the zero diagonal block in Eq. (10) and obtain the Schur complement for the pressure,

$$\begin{bmatrix} \mathbf{H} & \mathbf{G} \\ \mathbf{0} & \mathbf{D}\mathbf{H}^{-1}\mathbf{G} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{D}\mathbf{H}^{-1}\mathbf{f} \end{bmatrix}. \tag{18}$$

Eq. (18) can then be solved with a preconditioned Richardson iteration, for instance. Equivalently, we can factorize the block matrix in Eq. (10) in a block-LU decomposition,

$$\begin{bmatrix} \mathbf{H} & \mathbf{G} \\ \mathbf{D} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{D}\mathbf{H}^{-1} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{H} & \mathbf{G} \\ \mathbf{0} & \mathbf{D}\mathbf{H}^{-1}\mathbf{G} \end{bmatrix}, \tag{19}$$

and use the inverse of the lower triangular matrix as a transformation matrix for Eq. (10). In combination with a preconditioned Richardson iteration, this leads to a so-called *l-transforming iteration*, [44].

The block matrix in Eq. (18), or more precisely, the Schur complement matrix has normally exactly one zero eigenvalue (related to the pressure constant). If we ensure that the linear system of equations has at least one solution, we do not need to remove this singularity. Then, the employed iterative solution techniques are either able to handle such rank-deficient problems (Richardson iteration, multigrid with Gauss–Seidel smoothing) or do not lead to any complications in practice (BiCGstab[3]). In that sense, the matrix exponent $(\cdot)^{-1}$ does not refer to the inverse of a matrix (which might not exist) but indicates the application of a linear solver. The method which we use to ensure the solvability of Eqs. (10) and (18) will be described in Section 5.6.

Theoretically, we can find the pressure by solving the problem

$$\mathbf{A}\mathbf{p} = \mathbf{b}, \tag{20}$$

with $\mathbf{A} = \mathbf{D}\mathbf{H}^{-1}\mathbf{G}$ and the right-hand side $\mathbf{b} = \mathbf{D}\mathbf{H}^{-1}\mathbf{f}$. Once the pressure is found, the velocities can be determined from

$$\mathbf{H}\mathbf{u} = \mathbf{f} - \mathbf{G}\mathbf{p}. \tag{21}$$

Typically, both sub-problems are still too large for a direct solution and also a further reduction to smaller sub-problems is normally not practical.

Before we discuss the solution of Eqs. (20) and (21) we define the measure β for characterizing $\mathbf{H}$, because this operator appears in both equations and plays a major role in the iterative solution process,

$$\beta \equiv \frac{\Delta t}{2} \|\mathbf{L}\|_\infty \geqslant \frac{\Delta t}{2} \max_\Omega \left\{ \frac{1}{Re} \left(\hat{k}_{L,1}^2 + \hat{k}_{L,2}^2 + \hat{k}_{L,3}^2 \right) \right\}. \tag{22}$$

In practice, we find that the right-hand side of this inequality is almost equal to β which indicates that it scales like the temporal stability limit s_{visc} in Eq. (5). Therefore, a (semi-)implicit time integration can be more efficient, if the viscous time-step limit is more restricting than the convective limit, i.e. if

$$2\beta \gtrapprox s_{\mathrm{visc}}. \tag{23}$$

[3] In theory, the BiCGstab recurrence can break down before the exact solution is found even without this singularity. A modified algorithm to avoid such problems was proposed by Moriya et al. [33].

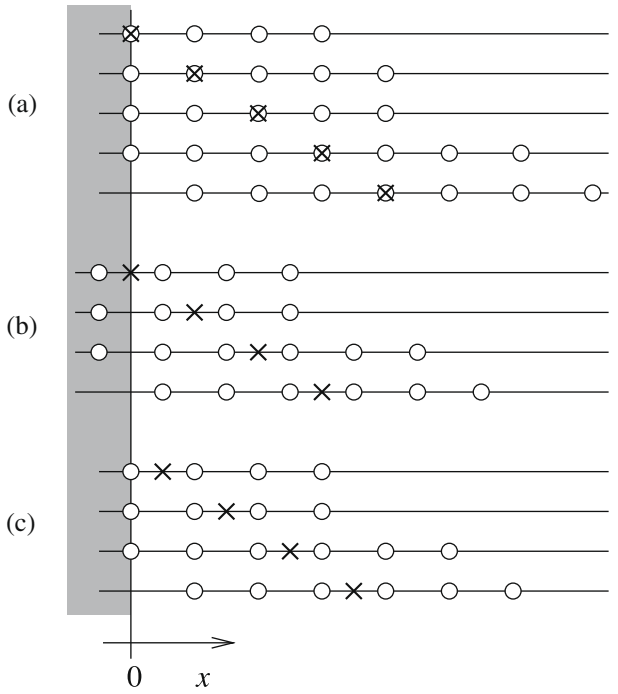

Fig. 6. Finite difference stencils of the d3 scheme near the boundary. Differentiation scenarios: (a) from a velocity grid to the same velocity grid, (b) from a velocity grid to the pressure grid and (c) from the pressure grid to a velocity grid.

Typically, the maximum modified wavenumber $\hat{k}$ of any spatial derivative increases with the convergence order and the stencil width n of the discretization. For central discretizations of $\mathbf{L}$, derived from Eq. (17), we find that $2 \leqslant \hat{k}_{L,i}\Delta x < \pi$, such that β is in the order of $\Delta t/\min_{\Omega,i}\{Re\Delta x^2\}$. Hence, large values of β correspond to a (locally) fine spatial resolution and/or to a coarse temporal resolution for a given Reynolds number Re.

5.2. Pressure iteration

The large matrix $\mathbf{A} = \mathbf{D}\mathbf{H}^{-1}\mathbf{G}$ in the pressure Eq. (20) is dense due to $\mathbf{H}^{-1}$, such that it cannot be stored explicitly. Because $\mathbf{A}$ is a Laplacian operator for $\beta = 0$ and has a zero eigenvalue anyway, the condition number $\kappa(\mathbf{A}) := |\lambda_{\max}(\mathbf{A})|/|\lambda_{\min}(\mathbf{A})|$ based on the eigenvalues $\lambda(\mathbf{A})$ is infinity. Hence, typical primary iterative solvers (such as Krylov subspace methods) will not work efficiently without an appropriate preconditioner. Luckily, there exist some efficient preconditioners $\widetilde{\mathbf{A}}$ (at least for certain ranges of β) such that the pressure equation can be solved in most cases with a moderate number of iterations with the preconditioned Richardson iteration scheme. It reads

$$\mathbf{p}^{l+1} = \mathbf{p}^{l} + \omega\widetilde{\mathbf{A}}^{-1}\mathbf{r}_A^{l}, \tag{24}$$

with a relaxation parameter ω to be defined and the residual

$$\mathbf{r}_A^{l} = \mathbf{b} - \mathbf{A}\mathbf{p}^{l} = \mathbf{D}\mathbf{H}^{-1}(\mathbf{f} - \mathbf{G}\mathbf{p}^{l}) \equiv \mathbf{D}\mathbf{u}^{l}. \tag{25}$$

The error in the pressure field is defined as

$$\mathbf{e}_A^{l} = \mathbf{p} - \mathbf{p}^{l} = \mathbf{A}^{-1}\mathbf{r}_A^{l}. \tag{26}$$

Because $\mathbf{e}_A^{l}$ is usually not accessible, the termination criterion for the iterative scheme is formulated for the residual,

$$\|\mathbf{r}_A^{l^*}\| \leqslant \epsilon_A, \tag{27}$$

with some threshold $\epsilon_A \geqslant 0$ and the corresponding iteration count l^*. The pressure iteration is illustrated in Fig. 7 (the details on the preconditioner $\widetilde{\mathbf{A}}$ are described in Section 5.2.1). As initial guess $\mathbf{p}^0$, we use the pressure field from the previous sub-time step, because the flow and pressure fields will change only very little between sub-time steps. Otherwise, a zero initial guess may be more favorable.

The Richardson iteration converges toward the exact solution if the spectral radius $\rho(\mathbf{I} - \omega\widetilde{\mathbf{A}}^{-1}\mathbf{A})$ is less than unity. Along the same line of thought, we see that the convergence ratio $\|\mathbf{r}_A^{l+1}\|_2/\|\mathbf{r}_A^{l}\|_2$ is bounded by $\rho(\mathbf{I} - \omega\widetilde{\mathbf{A}}^{-1}\mathbf{A})$ if $\widetilde{\mathbf{A}}^{-1}\mathbf{A}$ is Hermitian. This may not be the case if non-equidistant grids and/or high convergence orders at boundaries are employed, such that the

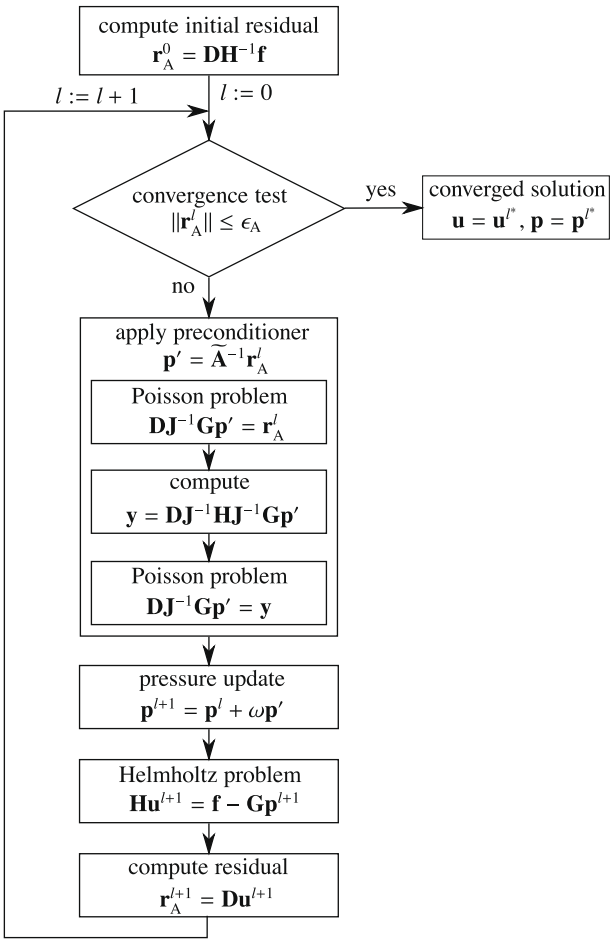

Fig. 7. Flow chart of the pressure iteration with the preconditioner (30). The vectors $\mathbf{p}'$ and $\mathbf{y}$ are only temporary variables in the context of the preconditioner.

spectral radius is only an estimate of the convergence ratio. These relations stress the importance of a good preconditioner $\widetilde{\mathbf{A}}$, i.e. a preconditioner which is "close" to the problem matrix $\mathbf{A}$. In addition to the convergence ratio, we define also the *convergence rate* as $-\log(\|\mathbf{r}^{l+1}\|/\|\mathbf{r}^l\|)$ for future use.

We assume for simplicity that the number of iterations to solve Eq. (20) depends primarily on the parameter β but not on the problem size or the degree of parallelization which. This assumption is actually supported by the results in Sections 7 and 8. Hence, the complexity to solve the equation is given by the complexity of the Richardson iteration (24) plus the complexities to apply $\mathbf{A} = \mathbf{DH}^{-1}\mathbf{G}$ and the preconditioner $\widetilde{\mathbf{A}}^{-1}$. The latter will be discussed in the following sections.

5.2.1. Preconditioner for the pressure iteration

The inverse of $\mathbf{A}$ is dense due to the elliptic problem character and the same will hold for a good preconditioner $\widetilde{\mathbf{A}}$. Therefore, $\widetilde{\mathbf{A}}$ must be a *forward-type* preconditioner [6] which means that we apply the preconditioner by solving at least one system of equations. This makes sense only if the cost for solving the preconditioner problem is much smaller than the cost for the unpreconditioned problem. Therefore, the preconditioner should be readily accessible and must not contain a dense matrix such as $\mathbf{H}^{-1}$.

It is straightforward to approximate $\mathbf{H}^{-1}$ in $\mathbf{A}$ with an explicitly accessible operator $\widetilde{\mathbf{H}}^{-1}$. Brüger et al. [5] choose $\widetilde{\mathbf{H}} = \mathbf{J}$ such that they obtain a Laplacian preconditioner,

$$\widetilde{\mathbf{A}} = \mathbf{DJ}^{-1}\mathbf{G}. \tag{28}$$

Obviously, this choice is only appropriate if β is small. In such situations, however, explicit time integration schemes may be more efficient anyway (as we will explain below). The well-known SIMPLE-type preconditioners [34] use improved approximations to $\mathbf{H}^{-1}$ (e.g., $\widetilde{\mathbf{H}}^{-1} = \mathrm{diag}\{\mathbf{H}\}^{-1}$) which are also easy to derive and have a sparse structure.

Other approaches try to approximate $\mathbf{A} = \mathbf{DH}^{-1}\mathbf{G}$ and not just $\mathbf{H}^{-1}$. Such a method has been proposed by Elman [11] for steady-state problems, where $\mathbf{H}$ is a convection–diffusion operator. In its simplest version it reads

$$\widetilde{\mathbf{A}} = \mathbf{DG}(\mathbf{DHG})^{-1}\mathbf{DG}. \tag{29}$$

Heuristically, this preconditioner is motivated by an attempt to commute $\mathbf{H}^{-1}$ and $\mathbf{G}$. Alternatively, we can rewrite $\mathbf{H}$ as $\mathbf{J}^{-1}\mathbf{JH}$ and commute $\mathbf{H}^{-1}\mathbf{J}^{-1}$ with $\mathbf{G}$ approximately which yields the preconditioner

$$\widetilde{\mathbf{A}} = \mathbf{DJ}^{-1}\mathbf{G}(\mathbf{DJ}^{-1}\mathbf{HJ}^{-1}\mathbf{G})^{-1}\mathbf{DJ}^{-1}\mathbf{G}. \tag{30}$$

The same result can be derived for $\mathbf{H} = \mathbf{HJJ}^{-1}$ and an approximate commutation of $\mathbf{J}^{-1}\mathbf{H}^{-1}$ with $\mathbf{G}$. The application of preconditioner (30) within the pressure iteration is depicted in Fig. 7.

Tests have shown that the commutation-based preconditioners (29) and (30) provide the largest flexibility of all mentioned preconditioners with respect to the value of β (see also Elman et al. [10] for steady-state problems). However, their application requires two solutions of Poisson problems with matrix $\mathbf{DJ}^{-1}\mathbf{G}$. In the present implementation, we apply only the preconditioner (30) because it is technically similar to the Laplacian preconditioner (28) and allows us to switch easily between them. Additionally, it will be demonstrated in Section 7 that the solution of two Poisson problems (rather than just one) does not lead to significant extra work for small β if an adaptive termination threshold for the Poisson problems is used. However, a more detailed analysis of the preconditioner is not in the scope of the present study.

It is a disadvantage of the SIMPLE-type preconditioners that they contain the Helmholtz matrix $\mathbf{H}$ and thus the sub-time step size $c_m\Delta t$ implicitly, such that they have to be stored separately for each sub-time step and need to be recomputed as soon as Δt changes. Unlike the SIMPLE-type methods, the commutation-based preconditioners (29) and (30) contain the Helmholtz matrix $\mathbf{H}$ explicitly which avoids these technical hurdles.

Problems with large β require more computational effort, since for any of the mentioned preconditioners $\widetilde{\mathbf{A}}/\omega$ is an increasingly poor approximation to $\mathbf{A}$ as β increases. At a certain point, the spectral radius $\rho(\mathbf{I} - \omega\widetilde{\mathbf{A}}^{-1}\mathbf{A})$ is larger than unity and the Richardson iteration diverges. For a given mesh width Δx and Reynolds number Re this can be avoided by choosing either a smaller ω or by reducing the time step size Δt to reduce β. As mentioned earlier, β scales like the stability limit s_{visc} in Eq. (5) which means that β can theoretically impose a "viscous" time step limit for (semi-)implicit schemes for a given ω. As a consequence, very small β indicate that a fully explicit time integration scheme may be more efficient overall because $\mathbf{H}$ equals $\mathbf{J}$ such that exactly one pressure iteration (24) already leads to the exact solution. Nevertheless, at least one Poisson problem has to be solved, generally.

The complexity to apply $\widetilde{\mathbf{A}}^{-1}$ equals the complexity to apply $(\mathbf{DJ}^{-1}\mathbf{G})^{-1}$ (cf. Section 5.2.2) plus the complexities of $\mathbf{D}$, $\mathbf{G}$, $\mathbf{H}$ and $\mathbf{J}^{-1}$ which were discussed in Section 3.2. Note that $\mathbf{J}^{-1}$ is trivial to compute directly since only the boundary conditions need to be inverted.

5.2.2. Solution of the Poisson problems in the preconditioner

The preconditioner (30) includes two Poisson sub-problems of the form

$$\mathbf{Kx} = \mathbf{h}, \tag{31}$$

with the Laplacian operator $\mathbf{K} = \mathbf{DJ}^{-1}\mathbf{G}$. It is solved iteratively (iteration count j) with the termination criterion $\|\mathbf{r}_K^j\| \leqslant \epsilon_K$. The finite difference stencils for $\mathbf{D}$, $\mathbf{J}$ and $\mathbf{G}$ derived from Eq. (17) are generally non-symmetric due to non-uniform grids and boundary conditions, such that $\mathbf{K}$ is non-symmetric either. However, $\mathbf{K}$ approximates the continuous negative-semidefinite operator $\mathcal{DG}$ whose eigenvalue spectrum is purely real. Hence, we find that the imaginary parts of the eigenvalues of $\mathbf{K}$ are at least *nearly* zero if the grid is moderately stretched. Their real parts are located within the interval $[-\|\mathbf{K}\|_\infty, \|\mathbf{K} - \text{diag}\{\mathbf{K}\}\|_\infty - \|\text{diag}\{\mathbf{K}\}\|_\infty] \approx [-\|\mathbf{K}\|_\infty, 0]$ which follows from Gerschgorin's theorem [13]. Additionally, the eigenvalues are more or less evenly distributed (depending on the symbols of $\mathbf{K}$) and scale with $1/\Delta x^2$. Generally, such an eigenvalue spectrum leads to a small convergence rate for the unpreconditioned Richardson iteration or Krylov subspace methods because of the unfavorable conditioning of $\mathbf{K}$ [13]. However, the distribution and scaling properties of the eigenvalues can be exploited by multilevel methods such as multigrid. The high-frequency parts of the solution, corresponding to large absolute eigenvalues, can be found very efficiently by so-called *smoothers* such that successive smoothing, restriction and prolongation operations allow an effective treatment of all parts of the solution. The Gauss–Seidel iteration is known to be the most efficient smoother in this context.

In the present implementation, we solve Eq. (31) with the Krylov subspace method BiCGstab [42] with *right preconditioning*. We use a geometric multigrid scheme as a preconditioner, such that $\widetilde{\mathbf{K}}^{-1}$ stands for one application of multigrid with a V(m,m)-cycle [14] at a grid coarsening factor of two in all spatial directions. The variable m indicates the number of smoothing sweeps on each grid level. The convergence rate is limited by the efficiency of the smoothers and by the accuracy of the restriction and prolongation operators. The restriction is performed by injection and the prolongation operators are second-order accurate. It is usually sufficient to use only a second-order discretization within multigrid since the error caused by the different discretizations of $\widetilde{\mathbf{K}}$ and $\mathbf{K}$ is normally only a small part of the total approximation error of $\widetilde{\mathbf{K}}^{-1}$ with respect to $\mathbf{K}^{-1}$. On the coarser grid levels this discretization error is small anyway. Also the weak diagonal dominance of the square matrices is ensured with this discretization. Algebraic multigrid can be an attractive alternative because it does not require the explicit definition of coarser grids (aggregation) such that it is more convenient to apply in the case of more complex grids. However, the direct grid definition in the geometric version can lead to a larger convergence rate with respect to the number of floating-point operations. Additionally, with a Cartesian grid and a geometric multigrid method the present implementation requires only a minimum of data storage which is also beneficial for the performance on typical microprocessors.

Since small wavelengths of the solution require sufficiently coarse grids it is crucial for the solver performance that these grids are provided in the implementation. This is often problematic on parallel architectures since the communication time on these coarse grids increases with the number of processors as discussed in Section 3.2 for torus networks. For maximum efficiency and to yield the complexity of the coarse grids given in Section 3.2 we need to redistribute the coarse-grid problems on the network during the V-cycles in order to minimize the communication distance.

A good choice for a smoother is important for the performance of the multigrid preconditioner. Under certain conditions Chebychev smoothers are competitive with ordinary Gauss–Seidel smoothers in terms of convergence rates [1]. Such polynomial approaches allow a simple and straightforward parallel implementation but are not free of tuning parameters. Also the computational effort can be expected to be larger than for Gauss–Seidel smoothers in our application. In this work, we use either *processor-block Gauss–Seidel* smoothing [17], red-black ordered Gauss–Seidel or combinations of both. Processor-block Gauss–Seidel is not a genuine Gauss–Seidel method because lexicographically ordered Gauss–Seidel smoothers run separately on each of the sub-domains which are coupled only in a Jacobi-like fashion. For small numbers of grid points per sub-domain and large numbers of processors the Jacobi characteristics dominate and vice versa. If the grid is strongly anisotropic, i.e. if the grid spacings Δx vary significantly in at least one direction, standard splittings such as the Jacobi or the Gauss–Seidel method are likely to fail because smoothing of the solution in other directions with larger Δx becomes inefficient. Therefore, we treat strongly stretched grid lines implicitly within each processor-block to accelerate the convergence. The procedure is often termed *line relaxation* and may be applied in alternating directions, if necessary.

Because the residual $\mathbf{r}_A^l$ in Eq. (24) is usually unrelated to the residual from the previous iteration $\mathbf{r}_A^{l-1}$, a zero initial guess appears to be the best choice for the first Poisson problem in Eq. (30). The solution of the second Poisson problem is close to the solution of the first (especially if β is small) such that the latter can be used as an initial guess for the second problem.

The number of iterations to solve Eq. (31) with BiCGstab and multigrid is typically of order one and does not depend on the problem size or the degree of parallelization. Hence, the total complexity to compute $\mathbf{K}^{-1}\mathbf{h}$ is given by the complexities of $\mathbf{K}$, the contributions of the primary BiCGstab solver and the multigrid preconditioner which were discussed in Section 3.2.

5.3. Helmholtz problems

Since the computation of the residual $\mathbf{r}_A^l$ requires the solution of

$$\mathbf{H}\mathbf{u}^l = \mathbf{f} - \mathbf{G}\mathbf{p}^l, \tag{32}$$

with the velocity $\mathbf{u}^l$ as an intermediate result, a separate solution of Eq. (21) is usually not necessary once the residual of the pressure equation is sufficiently small. Eq. (32) is solved iteratively and terminated after $k = k^*$ iterations when the residual

$$\mathbf{r}_H^{l,k} = \mathbf{f} - \mathbf{G}\mathbf{p}^l - \mathbf{H}\mathbf{u}^{l,k} \tag{33}$$

satisfies the termination criterion

$$\|\mathbf{r}_H^{l,k^*}\| \leqslant \epsilon_H, \tag{34}$$

with the threshold $\epsilon_H \geqslant 0$.

The continuous operator $\mathcal{H}$ is positive definite and has a purely real eigenvalue spectrum in which the smallest eigenvalue is exactly one. Because the discrete operator $\mathbf{H}$ has nearly the same properties (if the grid is only moderately stretched), its condition number is bounded by $\kappa(\mathbf{H}) \leqslant \kappa_{\max} \approx 1 + \beta$. Because β is sufficiently small in practice, we can solve Eq. (32) with the unpreconditioned Krylov subspace method BiCGstab [42] which converges fast for small condition numbers. In cases where β is large, the Helmholtz problems tend to have Poisson problem character such that they can be treated similarly as the Poisson problems in the previous section. In terms of computational cost, the solution of the Helmholtz problems is typically equally or less expensive than the application of the preconditioner $\widetilde{\mathbf{A}}$.

Since the flow field $\mathbf{u}$ from the previous sub-time step is usually closer to the solution of the present sub-time step than the zero initial guess, it is mostly the better initial guess for the first instance of the Helmholtz problem in the pressure iteration (24). All later solutions during the pressure iteration ($l \geqslant 1$) are related to the previous solution by $\mathbf{u}^{l+1} = \mathbf{u}^l + \mathbf{H}^{-1}\mathbf{G}(\mathbf{p}^l - \mathbf{p}^{l+1})$ such that the difference between $\mathbf{u}^{l+1}$ and $\mathbf{u}^l$ depends only on the convergence rate of the pressure iteration. Therefore, the previous solution is usually the best initial guess.

Because the eigenvalue spectrum of $\mathbf{H}$ depends strongly on β, the number of iterations to solve Eq. (32) with BiCGstab to a given accuracy level will mostly depend on β but not on the problem size or the degree of parallelization. Hence, the complexity to solve the equation is given by the complexity of $\mathbf{H}$ and the contributions of the BiCGstab solver, cf. Section 3.2.

5.4. Relations between the termination criteria and the solution accuracy

Even if Eqs. (20) and (21) are solved exactly, the accuracy of the flow field $\mathbf{u}$ is limited by the error of the discretization scheme. In addition, the iterative solver creates an *iteration error* which can be limited by setting appropriate termination criteria. For efficiency reasons, this error should not be required to be much smaller than the discretization error.

The absolute iteration error $\mathbf{e}_{it}$ of the velocity $\mathbf{u} = \mathbf{H}^{-1}(\mathbf{f} - \mathbf{G}\mathbf{p})$ can be written as

$$\mathbf{e}_{it} = \mathbf{e}_H + \mathbf{e}_P + \mathbf{e}_{HP}, \tag{35}$$

where $\mathbf{e}_H$ is caused by the inexact solution of the Helmholtz problems, $\mathbf{e}_P$ by the inexact pressure and the remainder, $\mathbf{e}_{HP}$, multiplicatively by both, the inexact Helmholtz problems and the inexact pressure (for the ease of writing we omit the iteration indices $k,\ l$). Further, it follows from the divergence condition $\mathbf{Du} = \mathbf{0}$ and Eq. (25) that the divergence error is equal to the residual of the pressure problem such that $\mathbf{r}_A = \mathbf{De}_{it} = \mathbf{D}(\mathbf{e}_H + \mathbf{e}_P + \mathbf{e}_{HP})$. Provided that we are able to find the pressure exactly, the residual of the continuity constraint is still $\mathbf{r}_A = \mathbf{De}_H = \mathbf{DH}^{-1}\mathbf{r}_H$. Hence, we cannot expect in general that the residual $\|\mathbf{r}_A\|$ of the pressure equation can be reduced below

$$\epsilon_{A,\min} = \sup_{\|\mathbf{r}_H\| \leqslant \epsilon_H} \|\mathbf{DH}^{-1}\mathbf{r}_H\| \tag{36}$$

for a given threshold ϵ_H. In other words, ϵ_H must be chosen sufficiently small to allow the pressure iteration to reach the desired accuracy level ϵ_A, i.e. we have to require that at least

$$\epsilon_A \geqslant \epsilon_{A,\min}. \tag{37}$$

Because it is difficult to compute $\epsilon_{A,\min}$, we use instead (with consistent matrix norms $\|\cdot\|$)

$$\epsilon_A = \|\mathbf{D}\|\,\|\mathbf{H}^{-1}\|\epsilon_H \tag{38}$$

and tolerate the lower accuracy of the overall solution. In general, $\|\mathbf{H}^{-1}\|$ is not easy to compute either. At least for directly applied Dirichlet boundary conditions and the infinity norm $\|\cdot\|_\infty$ we know by design that each row sum of $\mathbf{H}^{-1}$ must be exactly one. For discretizations with a stencil width of $n = 3$ the matrix $\mathbf{H}^{-1}$ is positive because $\mathbf{H}$ is then a *M-matrix* [12]. Unfortunately, the Dirichlet boundary conditions involve interpolations such that the corresponding rows of $\mathbf{H}^{-1}$ inevitably contain elements with alternating sign. Hence, we conclude that $\|\mathbf{H}^{-1}\|_\infty \geqslant 1$. There is no indication that β has a significant influence on $\|\mathbf{H}^{-1}\|_\infty$ and we will also find from the numerical experiments in Section 6.2 that $\|\mathbf{H}^{-1}\|_\infty$ is of order one. For boundary conditions other than Dirichlet such an estimate is more difficult.

Next, we try to relate the thresholds ϵ_H and ϵ_A to the absolute iteration error $\mathbf{e}_{it}$. Normally, we can assume that the mixed error $\mathbf{e}_{HP}$ is much smaller than $\mathbf{e}_H$ and $\mathbf{e}_P$, such that Eq. (35) can be approximated by

$$\mathbf{e}_{it} \approx \mathbf{e}_H + \mathbf{H}^{-1}\mathbf{Ge}_A, \quad \mathbf{e}_{HP} \ll \min(\mathbf{e}_H, \mathbf{e}_P). \tag{39}$$

If we use Eq. (38) to relate ϵ_A to ϵ_H we can bound the absolute velocity-error by

$$e_{it} = \|\mathbf{e}_{it}\| \lessapprox \epsilon_H\|\mathbf{H}^{-1}\|(1 + \|\mathbf{D}\|\,\|\mathbf{H}^{-1}\mathbf{GA}^{-1}\|). \tag{40}$$

Numerical tests (see Section 6.2) and scaling arguments indicate that $\|\mathbf{D}\|_\infty\|\mathbf{H}^{-1}\mathbf{GA}^{-1}\|_\infty$ scales roughly like $1/\Delta x$ for Dirichlet boundary conditions. In practice, we estimate $\|\mathbf{H}^{-1}\|$ and $\|\mathbf{H}^{-1}\mathbf{GA}^{-1}\|$ such that we can determine ϵ_H and ϵ_A for the desired value of e_{it} according to Eqs. (40) and (38), respectively.

In contrast to our approach, Brüger et al. [5] formulate a termination criterion for the residual for the entire problem (10) which in our notation can be written as $\|(\mathbf{r}_H^T\mathbf{r}_A^T)^T\| \leqslant \epsilon$. However, this approach does not take any efficiency considerations into account, since the continuity constraint and the momentum equations are treated equally in this criterion such that the pressure might be solved more accurately than necessary. To ensure the convergence of the pressure iteration they require $\epsilon_H = \epsilon_K \leqslant C\epsilon$, where the constant C depends only on the spectral radius of the Richardson iteration for the entire problem (10).

5.5. Termination criterion for the preconditioner

The solution of the preconditioning problem,

$$\widetilde{\mathbf{A}}\Delta\mathbf{p}^{l+1} = \omega\mathbf{r}_A^l, \tag{41}$$

with $\Delta\mathbf{p}^{l+1} = \mathbf{p}^{l+1} - \mathbf{p}^l$ is typically the most time-consuming operation in the pressure iteration. Because the preconditioner $\widetilde{\mathbf{A}}/\omega$ is only an approximation to $\mathbf{A}$, it is not necessary to compute $\Delta\mathbf{p}^{l+1}$ up to machine precision. The error of the preconditioning problem, $\underset{\sim}{\mathbf{e}}_A^{l+1} = \omega\widetilde{\mathbf{A}}^{-1}\mathbf{r}_A^l - \Delta\mathbf{p}^{l+1}$, increases the error $\mathbf{e}_A^{l+1}$ of the pressure and reduces the convergence rate of the pressure iteration. Therefore, $\underset{\sim}{\mathbf{e}}_A^{l+1}$ should be sufficiently small compared to $\mathbf{e}_A^{l+1}$.

In practice, we assume $\underset{\sim}{\mathbf{r}}_A^{l+1} \approx \mathbf{r}_K^{l+1}$ and require that $\epsilon_K^{l+1} = \phi\|\mathbf{r}_A^{l+1}\|$ with a relaxation factor $\phi < 1$. Unfortunately, $\|\mathbf{r}_A^{l+1}\|$ is not known beforehand, but we can estimate it by extrapolating the residuals of the previous time levels,

$$E\left(\|\mathbf{r}_A^{l+1}\|_{t,m}\right) = \sum_{i=1}^{s} w_i\|\mathbf{r}_A^{l+1}\|_{t-i\Delta t,m}, \quad \text{with } \sum_{i=1}^{s} w_i = 1. \tag{42}$$

The extrapolation weights w_i can be found from Eq. (17) with $\Delta t < 0$ in place of Δx. In the present implementation, we simply use $s = 1$ and $w = 1$. This extrapolation has to be done separately for each Runge–Kutta sub-time step m, because $\|\mathbf{r}_A^{l+1}\|_{t,m}$ can be expected to be more or less smooth in $t, t + \Delta t, t + 2\Delta t, \ldots$, but not in m. Then, the termination threshold for the preconditioner at time t, sub-time step m and iteration $l + 1$ is

$$\epsilon_{K,t,m}^{l+1} = \phi E\left(\|\mathbf{r}_A^{l+1}\|_{t,m}\right). \tag{43}$$

In practice, we find that ϕ does not have to be much smaller than unity to obtain a good performance of the overall iteration method. Especially in the case of coarsely resolved and unsteady flows, the convergence rates in the pressure iteration will change more rapidly such that smaller ϕ will adapt the termination threshold faster to the actual situation. Numerical tests show that values between 0.1 and 1.0 are good choices for ϕ. They do not need to be smaller because it can be cheaper overall to tolerate a few more pressure iterations (because of a large ϕ) than solving the preconditioner problem fewer times but more accurately. Different to our approach, Brüger et al. [5] directly set $\epsilon_K = \epsilon_H$ such that the preconditioner problems are normally solved too accurately.

5.6. Solvability

The singular system (10) has only a solution if the right-hand side is fully contained in the column space of the system matrix. In this case the rank deficiency of the matrix is normally not a problem for the iterative solvers employed in this work (cf. Section 5.1). A right-hand side which is not fully contained in the column space of the matrix indicates that the boundary conditions try to enforce a net increase or decrease of mass which violates the mass conservation law. Unfortunately, this is often the case, e.g. due to discretization errors at the boundary [5] or convective outflow boundary conditions [38].

There exist two different methods to resolve this problem: the easiest way is to prescribe the pressure artificially at least one given point in space (which corresponds to a Dirichlet boundary condition for the pressure). Then $\mathbf{A}$ becomes non-singular and a solution always exists. The disadvantage of this method is that it replaces the governing equation at these grid points by an artificial pressure constraint, such that the flow field is generally not divergence-free there. Additionally, the solution is normally not smooth in these areas which can lead to stability problems especially in the case of large numbers of grid points. We can interpret these points as mass sinks (or sources) which compensate for the net outflow (or inflow) due to the boundary conditions.

In the present implementation an alternative method is employed which is similarly described in Simens et al. [38] for a fractional step method. Rather than modifying the system matrix, the right-hand side $\mathbf{b}$ is corrected such that it lies in the column space of $\mathbf{A}$. Once a solution for the pressure is found, the undefined part of the solution (the absolute pressure level) can be chosen freely by adding a constant to $\mathbf{p}$. The left null space of $\mathbf{A}$ is the orthogonal complement to its column space. It has rank one and can be represented by an arbitrarily scalable vector $\boldsymbol{\psi}$ which satisfies $\boldsymbol{\psi}^T\mathbf{A} = \mathbf{0}$. The right-hand side $\mathbf{b}$ must then be orthogonal to $\boldsymbol{\psi}$ because

$$\boldsymbol{\psi}^T\mathbf{A}\mathbf{p} = \boldsymbol{\psi}^T\mathbf{b} = 0. \tag{44}$$

With

$$\boldsymbol{\phi} = \mathbf{H}^{-T}\mathbf{D}^T\boldsymbol{\psi}, \tag{45}$$

the right-hand side $\mathbf{f}$ of the Helmholtz problem (21) must be orthogonal to $\boldsymbol{\phi}$, i.e.,

$$\boldsymbol{\phi}^T\mathbf{f} = 0. \tag{46}$$

To satisfy Eq. (46), $\mathbf{f}$ is corrected to $\mathbf{f}_{\text{corr}}$ by projecting it along a vector onto the column space of $\mathbf{A}$, i.e.,

$$\mathbf{f}_{\text{corr}} = \mathbf{f} - \frac{\boldsymbol{\phi}^T\mathbf{f}}{\boldsymbol{\phi}^T\boldsymbol{\theta}}\boldsymbol{\theta} \quad \text{with } \boldsymbol{\phi}^T\boldsymbol{\theta} \neq 0. \tag{47}$$

Then $\boldsymbol{\psi}^T\mathbf{f}_{\text{corr}} = 0$ is satisfied and the pressure equation has at least one solution. The projection vector $\boldsymbol{\theta}$ can be chosen freely as long as it satisfies the restriction in Eq. (47). If, for instance, the boundary conditions shall be corrected only for one velocity component at one grid point the vector $\boldsymbol{\theta}$ is zero except at one entry. If the 2-norm of the correction, $\|\mathbf{f}_{\text{corr}} - \mathbf{f}\|_2$, shall be minimal we choose $\boldsymbol{\theta} = \boldsymbol{\phi}$ such that the correction Eq. (47) is an orthogonal projection of $\mathbf{f}$ onto the column space of $\mathbf{A}$.

The vector $\boldsymbol{\psi}$ can be computed using the same methods as for solving $\mathbf{A}\mathbf{p} = \mathbf{b}$. Because $\mathbf{A}^T$ does normally not have the character of a partial differential equation, the application of geometric multigrid preconditioning can be difficult. The computation of $\boldsymbol{\phi}$ according to Eq. (45) is not problematic because preconditioning of the $\mathbf{H}^T$ operator is usually not necessary, similarly to $\mathbf{H}$. However, $\mathbf{H}$ is different for each sub-time step and changes with the time step size, such that $\boldsymbol{\phi}$ (or at least $\boldsymbol{\psi}$) must be stored for each sub-time step and recomputed if the time step size changes. If this happens only a few times during the simulation, extra costs due to an inefficient application of the multigrid method can be tolerated. In the case of a purely explicit time integration scheme, $\mathbf{H}$ is equal to $\mathbf{J}$ and $\boldsymbol{\phi}$ is the same for all sub-time steps and all times.

The discussed solvability problem also applies to the sub-problems (31) inside the preconditioner and the same measures have to be applied accordingly.

6. Validation

In this section, different aspects of the present implementation are checked for their correctness. For this purpose, we define an absolute error e which compares the numerically found velocity $\mathbf{u}$ to the exact solution $\bar{\mathbf{u}}$,

$$e = \|\mathbf{u} - \bar{\mathbf{u}}\|_\infty = e_{it} + e_d. \tag{48}$$

In this context, e_d is the absolute discretization error and e_{it} is the absolute error due to the iterative solution. In addition, we use only the infinity norm $\|\cdot\|_\infty$ for all matrix and vector norms.

All simulations are carried out for a periodic straight channel. The code is implemented in FORTRAN90 and uses the *Message Passing Interface* (MPI). The relaxation factors are set to $\omega = 1$ and $\phi = 0.1$. The spatial coordinates $x_1,\ x_2,\ x_3$ point in the streamwise, wall-normal and spanwise direction, respectively. The reference values for the Reynolds number Re are the maximum streamwise velocity at $t = 0$ and the channel half-height. The grids are equidistant in the lateral directions and only in the wall-normal direction the grid points are stretched according to

$$x_{2,i} = 1 - \frac{\cos(\kappa(\xi + i - 1))}{\cos(\kappa\xi)} \quad \text{with } i = 1, 2, \ldots, N \quad \text{and} \quad \kappa = \frac{\pi}{N - 1 + 2\xi} \tag{49}$$

for different values of ξ (N denotes the number of grid points of the pressure grid). The grid is equidistant for $\xi \to \infty$ and for $\xi = 0$ we obtain the Gauss–Lobatto points.

The iteration thresholds ϵ_H and ϵ_A are coupled according to Eq. (38) such that only one of them needs to be specified. Because only Dirichlet boundary conditions are employed and we do not have a better estimate at hand, we simply assume that $\|\mathbf{H}^{-1}\|$ equals one though it is actually larger but still of order one. This is caused by the interpolations which are required to enforce the Dirichlet boundary conditions, cf. Section 5.4. Anyway, we need to satisfy Eq. (37) which is not violated with our choice.

6.1. Convergence order

The convergence order of the discretization is tested in time and space by simulating a two-dimensional flow field in a channel with the dimensions $L_1 \times L_2 = 2 \times 2$. It is given by

$$\bar{u}_1(x_1, x_2, t) = \frac{1}{2}[\sin(k_1 x_1) + 1]\sin(k_2 x_2) e^{-\sigma t}\frac{k_2}{k_1}, \tag{50a}$$

$$\bar{u}_2(x_1, x_2, t) = \frac{1}{2}[\cos(k_2 x_2) - 1]\cos(k_1 x_1) e^{-\sigma t}, \tag{50b}$$

$$\bar{u}_3(x_1, x_2, t) = 0. \tag{50c}$$

In this section we choose the wave numbers $k_1 = k_2 = \pi$ such that the maximum amplitudes of u_1 and u_2 are equal to one. The corresponding pressure is determined from Eq. (3),

$$\bar{p}(x_1, x_2, t) = \left\{ \frac{1}{16}[\cos(2\pi x_1) - \cos(2\pi x_2)] + \frac{1}{4}[\cos(\pi x_2) - \sin(\pi x_1) + \sin(\pi x_1)\cos(\pi x_2))] \right.$$
$$\left. - \frac{1}{20}[\sin(\pi x_1)\cos(2\pi x_2) + \cos(2\pi x_1)\cos(\pi x_2)] \right\} e^{-2\sigma t}. \tag{51}$$

This flow field has zero divergence, but the momentum Eq. (1a) is only satisfied if the residual

$$\mathcal{R} = \frac{\partial \bar{u}}{\partial t} - \mathcal{N}(\bar{u}) - \mathcal{L}\bar{u} + \nabla \bar{p} \tag{52}$$

is added as a forcing term to its right-hand side. Note that $\mathcal{R}$ needs to be computed at each Runge–Kutta sub-time step, cf. Section 4.1.

The investigated parameter settings are listed in Table 4 with N_t as the number of time steps and $N_1,\ N_2$ as the numbers of grid points in the two spatial directions. The grid is equidistant ($\xi = \infty$) and the termination threshold $\epsilon_H = 10^{-14}$ is set close to machine accuracy such that $e \approx e_d$.

6.1.1. Spatial convergence order

The spatial convergence properties are assessed by varying the number of grid points in both spatial directions simultaneously. The time dependence is eliminated by choosing $\sigma = 0$ such that no time integration error can occur. Because the convective and the viscous terms in Eq. (1a) should have the same order of magnitude in this test (with respect to k_1 and k_2) we set $Re = 10$. With $\Delta t = 10^{-4}$ the number of iterations in the Richardson scheme (24) is about three to five.

Fig. 8 shows the absolute error e for the cases A and B. An inspection of the flow field indicates that the largest error in both cases originates from grid points near the boundary. Because e measures the amplitude error, it grows like Δx^4 (case A) and Δx^6 (case B), respectively. These convergence rates are equal to the smallest convergence rates in the respective discretization schemes (cf. Table 3). If the phase errors were considered, we would expect the error to scale only like Δx^3 and Δx^5 (at least for sufficiently small Δx). In case B, the error scales like Δx^{-1} for very small Δx and reaches a minimum of about $5 \cdot 10^{-11}$. This is probably related to small stencil coefficients which lead to a loss of significant digits. In addition, the matrix in Eq. (16) is increasingly ill-conditioned with growing convergence order such that the accuracy of the finite difference coefficients is limited.

Table 4
Parameters for convergence tests and tests of the termination criteria ($L_1 \times L_2 = 2 \times 2$; equidistant grids, $\xi = \infty$; discretization schemes from Table 3).

Case	Re	σ	Δt	N_t	$(N_1+1), N_2$	Discretization
A	10	0	10^{-4}	1	9 ... 2 049	d3
B	10	0	10^{-4}	1	17 ... 2 049	d5
C	1	100	$2.44 \cdot 10^{-6} \ldots 10^{-2}$	1 ... 4 096	129	d5
D	10	100	$2.44 \cdot 10^{-6} \ldots 10^{-2}$	1 ... 4 096	129	d5
E	1 000	100	$2.44 \cdot 10^{-6} \ldots 10^{-2}$	1 ... 4 096	129	d5
F	10	10	10^{-4}	1	257	d5
G	10	10	10^{-4}	1	65	d5

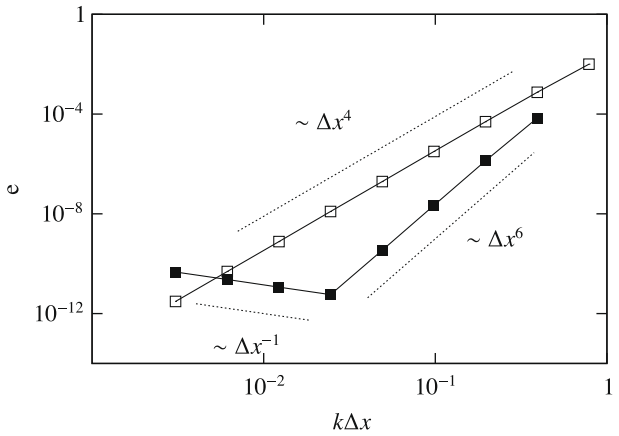

Fig. 8. Absolute error e for cases A (□) and B (■).

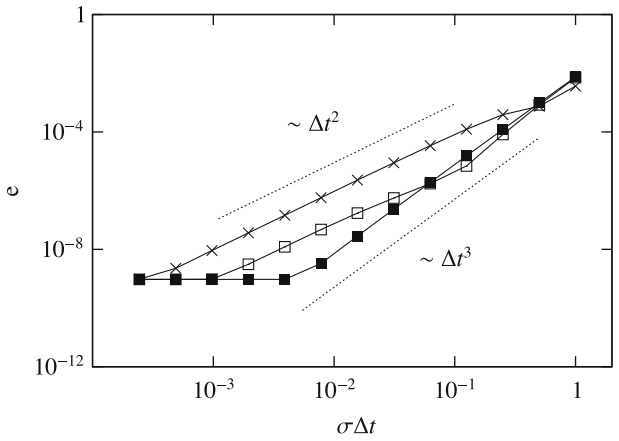

Fig. 9. Absolute error e for cases C (×), D (□) and E (■).

6.1.2. Temporal convergence order

A direct measurement of the time integration error requires that the spatial discretization error is negligibly small. We achieve this with a high spatial convergence order in combination with relatively large time steps. In general, the time integration method (7) is only second-order accurate because the viscous terms are integrated with the Crank–Nicolson scheme while the convective terms are integrated with a third-order accurate Runge–Kutta scheme. It depends on the magnitude of $Re\Delta t$ whether the error is dominated by the convective or the viscous terms. This is demonstrated in Fig. 9 for the cases C, D and E which differ only in their Reynolds number. For $Re = 10$ (case D) the maximum error is dominated by the viscous terms for small Δt and by the convective terms for large Δt. The discretization error $e_d \approx e$ scales like Δt^2 and Δt^3, respectively. If the Reynolds number is sufficiently large (as in case E with $Re = 1000$) the accuracy in the investigated range of time step sizes is

Table 5
Parameters for the numerical experiments to test the termination criteria of the solver (cf. Fig. 10).

Experiment	Pressure problems	Helmholtz problems	ϵ_{var}	ϵ_{ref}
1 (+)	$\epsilon_A = \|\mathbf{H}^{-1}\|\,\|\mathbf{D}\|\epsilon_{\text{ref}}$	$\epsilon_H = \epsilon_{\text{var}}$	$10^{-14}\ldots10^{-4}$	10^{-14}
2 (□)	$\epsilon_A = \|\mathbf{H}^{-1}\|\,\|\mathbf{D}\|\epsilon_{\text{var}}$	$\epsilon_H = \epsilon_{\text{ref}}$	$10^{-14}\ldots10^{-4}$	10^{-14}
3 (■)	$\epsilon_A = \|\mathbf{H}^{-1}\|\,\|\mathbf{D}\|\epsilon_{\text{var}}$	$\epsilon_H = \epsilon_{\text{var}}$	$10^{-14}\ldots10^{-4}$	n/a

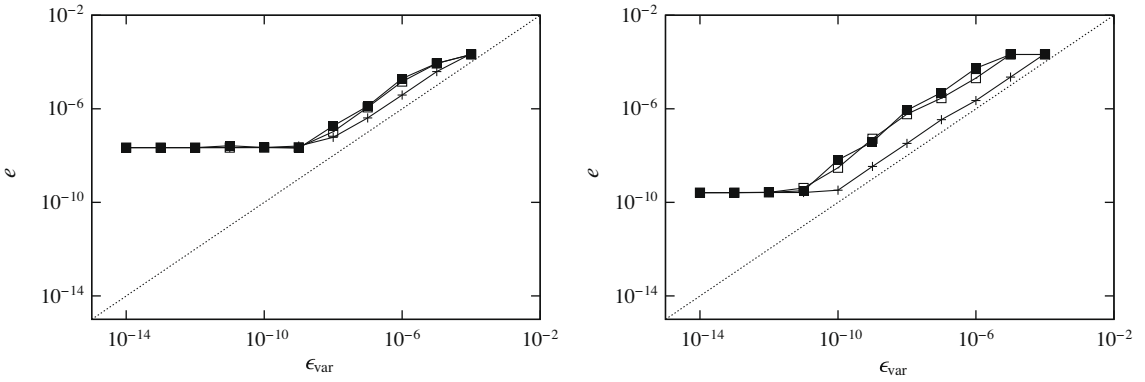

Fig. 10. Absolute error e with respect to the termination criteria ϵ_H and ϵ_A for first (+), second (□) and third (■) experiment (left: case F, right: case G).

dominated by the convective terms leading to a third-order convergence rate. A purely second-order convergence rate can be found for $Re = 1$ (case C).

6.2. Termination criteria and solution accuracy

In this section, the influence of the termination criteria ϵ_H and ϵ_A on the absolute error e is tested. We use the same flow configuration as in the previous section. In this test, it is necessary that the residual norm meets the desired level of accuracy as closely as possible, i.e. we try to satisfy $\|\mathbf{r}\| = \epsilon - \delta$ with $0 \leqslant \delta < \epsilon$ where $\delta \leqslant \|\mathbf{r}^{l^r-1}\| - \|\mathbf{r}^{l^r}\|$ should be as small as possible. Therefore, the convergence rates must be kept very small such that the solution is approached in small steps. We achieve this for the pressure iteration by using the Laplacian preconditioner (28) and by terminating the solver for the preconditioning problem already when its residual has reached half the size of the current residual of the pressure iteration, i.e. $\|\mathbf{r}_K^{l+1}\| \leqslant 0.5\|\mathbf{r}_A^l\|$. The preconditioning problem uses only the plain BiCGstab solver (without multigrid preconditioning). Similar measures are taken for the Helmholtz problems where we replace the BiCGstab solver by a plain Richardson iteration without preconditioning. A zero vector is used as initial guess for all solvers.

As explained in Section 5.4, the termination threshold ϵ_H for the Helmholtz problems implicitly defines the termination threshold ϵ_A for the pressure equation according to Eq. (38). To test this relation between ϵ_H and ϵ_A, the termination criteria are varied independently in the following three numerical experiments (see also Table 5).

In the first experiment, we vary the termination threshold ϵ_H and solve the pressure equation very accurately by setting $\epsilon_A = \|\mathbf{H}^{-1}\|\,\|\mathbf{D}\|\epsilon_{\text{ref}}$ where $\epsilon_{\text{ref}} = 10^{-14}$ is close to the machine precision. To obtain an accurate right-hand sides for the pressure equation we compute the velocities first with a strict termination threshold $\epsilon_H = \epsilon_{\text{ref}}$. Then, the velocities are computed a second time with varied $\epsilon_H = \epsilon_{\text{var}}$. The influence of ϵ_{var} on the absolute error e is shown in Fig. 10. As expected, we find that e is only slightly larger than $\epsilon_H = \epsilon_{\text{var}}$ if the discretization error e_d is not larger than the iteration error e_{it} for small ϵ_{var} and if the initial guess is not already accurate enough to satisfy the termination criterion for large ϵ_{var}. This indicates that $\|\mathbf{H}^{-1}\|$ is obviously of order one.

In the second experiment, we keep $\epsilon_H = \epsilon_{\text{ref}}$ close to machine precision and vary $\epsilon_A = \|\mathbf{H}^{-1}\|\,\|\mathbf{D}\|\epsilon_{\text{var}}$ instead. As shown in Fig. 10, the absolute error e is now much larger than ϵ_{var} in the same parameter range of ϵ_{var} as in the first experiment. Because arithmetic errors are of the same order as before, the large difference must be related to the value of $\|\mathbf{D}\|\,\|\mathbf{H}^{-1}\mathbf{G}\mathbf{A}^{-1}\|$. From the plots, we estimate that $\|\mathbf{D}\|\,\|\mathbf{H}^{-1}\mathbf{G}\mathbf{A}^{-1}\| \approx 15$ for case F and $\|\mathbf{D}\|\,\|\mathbf{H}^{-1}\mathbf{G}\mathbf{A}^{-1}\| \approx 60$ for case G which indicates that $\|\mathbf{D}\|\,\|\mathbf{H}^{-1}\mathbf{G}\mathbf{A}^{-1}\| \sim \Delta x^{-1}$ and $\|\mathbf{H}^{-1}\mathbf{G}\mathbf{A}^{-1}\| \sim \Delta x^0$.

In the third and final experiment, both termination criteria ϵ_H and ϵ_A are coupled according to Eq. (38) and varied simultaneously ($\epsilon_H = \epsilon_{\text{var}}$). Fig. 10 shows that the error is comparable to the second experiment, because the total iteration error is dominated by the pressure iteration and not by the residual of the Helmholtz problems. We also conclude that the coupling of ϵ_A and ϵ_H according to Eq. (38) is indeed not too strict with respect to ϵ_A, because otherwise the termination criterion (27) for the pressure iteration could only hardly be satisfied within a limited number of iterations. On the other hand, the

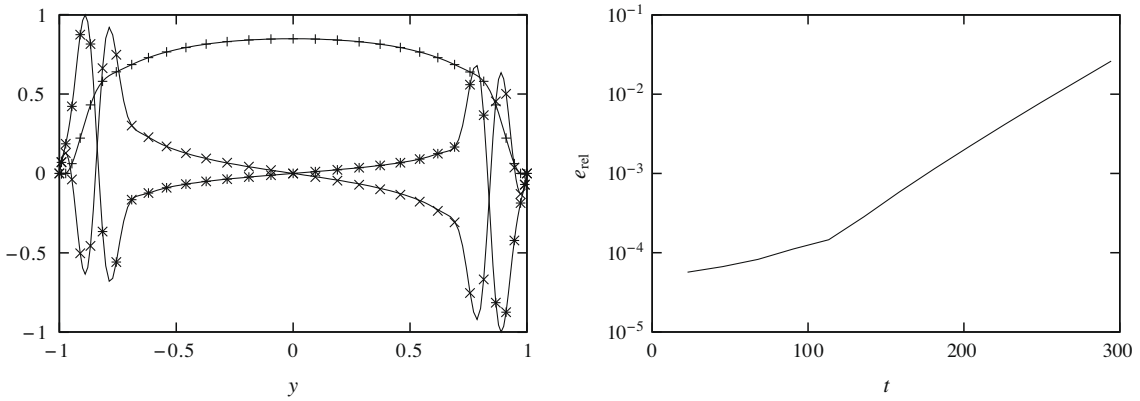

Fig. 11. Left: shape of the normalized eigensolution scaled by $\exp(-\lambda_i t)$. Lines: $t = 0$, points: $t = 226.5$ $(+u'_1, \times u'_2, *u'_3)$. Right: evolution of the relative error e_{rel}.

iteration errors in the second and third experiment could be further reduced with smaller values for ϵ_A closer to $\epsilon_{A,\min}$ (cf. Eq. (36)).

6.3. Orr–sommerfeld and squire modes for poiseuille flow

In a next validation test, we simulate the temporal evolution of a three-dimensional Orr–Sommerfeld and Squire eigenmode for plane Poiseuille flow. The stream- and spanwise wave numbers of the tested mode are $k_{1,3} = 2\pi/L_{1,3} = 1$ and the Reynolds number is set to $Re = 10000$. The eigenvalue is $\lambda = \lambda_r + i\lambda_i = 0.2774 - 0.02411\ i$ which indicates a temporally decaying solution. The extents of the physical domain are $L_1 \times L_2 \times L_3 = 2\pi \times 2 \times 2\pi$ which is discretized with the d3 scheme on $N_1 \times N_2 \times N_3 = 32 \times 129 \times 32$ grid points. The grid is non-uniform in the wall-normal direction according to Eq. (49) with $\xi = 10$ to obtain a more accurate representation of the eigenmode near the walls. The time step size is limited to $\Delta t = 0.01$ such that it is much smaller than its maximum value according to Eq. (4) and the time integration is more accurate. Nonlinear effects are negligible because the initial amplitude of the modal perturbation $u' = [u'_1, u'_2, u'_3]^T$ is very small with respect to the base flow $u_{\text{base}} = [1 - x_2^2, 0, 0]^T$, i.e. $\|\mathbf{u}'(t=0)\|/\|\mathbf{u}_{\text{base}}\| = 10^{-5}$. The termination threshold ϵ_H for the Helmholtz problems is adapted in each time step to the current disturbance magnitude according to $\epsilon_H(t) = 10^{-7}\|\mathbf{u}'(t)\|$ such that a sufficient accuracy of the iterative scheme is guaranteed at all times.

The shape of the perturbation is plotted in Fig. 11 for $t = 0$ and for $t = 10 \cdot 2\pi/\lambda_r = 226.5$. It shows clearly that the simulation yields the correct decay rate and that it maintains the shape of the eigenmode. Nevertheless, the relative error $e_{\text{rel}} = \|\mathbf{u}'(t=0) - \mathbf{u}'(t)\exp(-\lambda_i t)\|/\|\mathbf{u}'(t=0)\|$ of the solution increases in time because of the accumulation of temporal and spatial discretization errors which trigger the growing Tollmien–Schlichting mode (Fig. 11).

6.4. Transitional channel flow

Next, we simulate a temporal transition from laminar to turbulent channel flow to validate the present implementation for a more demanding flow configuration which involves strong nonlinear effects. To this end, we perform a direct numerical simulation at $Re = 5000$ $(Re_\tau = 208)$ in a domain with the dimensions $L_1 \times L_2 \times L_3 = 2\pi/1.12 \times 2 \times 2\pi/2.1$. Because the finite difference discretization is less accurate at high wave numbers than a spectral discretization which is used for comparison, we choose a finer resolution $(N_1 \times N_2 \times N_3 = 256 \times 257 \times 256)$ with the d3 discretization scheme. The grid is non-uniform in the wall-normal direction according to Eq. (49) with $\xi = 16$ to account for the higher resolution requirements near the walls. The time step size is adjusted every tenth time step such that $\Delta t \approx 0.5\Delta t_{\max}$ according to Eq. (4). The initial Poiseuille flow is perturbed by a two-dimensional stable Tollmien–Schlichting wave with a maximum amplitude of 3% of the laminar center-line velocity and two oblique three-dimensional stable modes with amplitude 0.1%. The wave lengths of the three perturbations are identical to the streamwise (and spanwise) extent of the domain and the bulk velocity is fixed to $u_b = 2/3$ for all times. For further details on the flow configuration see [37]. The transition from laminar to turbulent flow requires a sufficiently accurate representation of a large number of excited modes. A termination threshold of $\epsilon_H = 10^{-8}$ was found to be sufficiently accurate for the transitional phase $t = 0, \ldots, 200$, whereas for the turbulent phase $(t = 200, \ldots, 1000)$ we chose $\epsilon_H = 10^{-6}$. The simulation was carried out on a Cray XT3 supercomputer on $P = 8 \times 4 \times 8$ CPU cores.

We compare our results to flow fields obtained with a pseudospectral simulation code (Chebychev series in the wall-normal direction and Fourier series in the lateral directions) which uses $N_1 \times N_2 \times N_3 = 160 \times 161 \times 160$ grid points and the same CN-RK3 time integration scheme [37]. Note that the flow statistics in this reference simulation have been computed on a coarser grid with $32 \times 33 \times 32$ grid points. Fig. 12 shows that the evolution of Re_τ matches the reference results quite

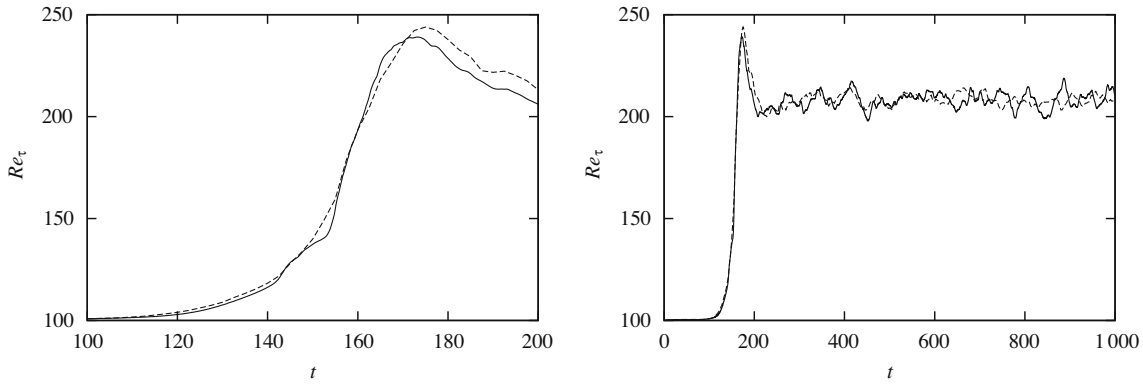

Fig. 12. Temporal evolution of Re_τ compared to the results from Schlatter [37] ((————) present simulation, (- - -) reference solution; left: initial phase of transition, right: full simulation).

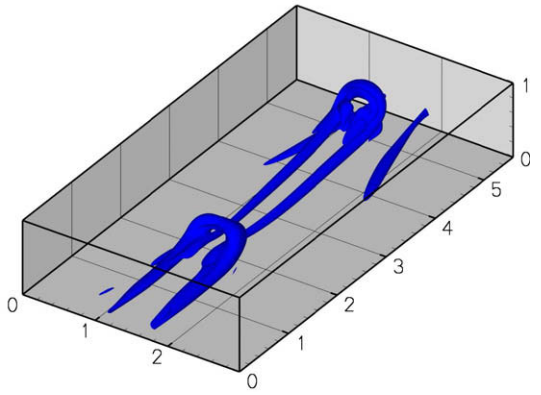

Fig. 13. Hairpin vortex at $t = 136$ (isosurface for $\lambda_2 = -0.1$ [20]).

well. Differences between the two simulations are visible especially during the streak breakdown ($t \approx 160, \ldots, 200$). They are mostly related to slightly different initial conditions (e.g. different relative phase shifts between the three disturbance modes). The formation of hairpin vortices is visualized in Fig. 13 which compares well to the plots in [37]. Finally, we examine different statistical measures for the turbulent flow field which are computed between $t = 500$ and 1000. The mean velocity profile (Fig. 14) matches the wall laws $u^+ = y^+$ and $u^+ = 2.5\log(y^+) + 5.5$ very well. Also the Reynolds stresses and the energy budget (turbulent production $\mathcal{P}$, viscous dissipation due to mean flow strain $\varepsilon_{\text{mean}}$, viscous dissipation due to fluctuations $\varepsilon_{\text{fluct}}$) are in close agreement with the reference results (Fig. 15).

7. Performance test of the preconditioners

To assess the preconditioner performance, we compare semi-implicit CN-RK3 time integration with explicit RK3 time integration with respect to the elapsed times to integrate one time unit of turbulent channel flow. In these tests we distinguish between the commutation-based preconditioner (30) and the Laplacian preconditioner (28). For explicit time integration the Helmholtz matrix simplifies to $\mathbf{H} = \mathbf{J}$, such that only one pressure iteration is necessary per sub-time step and the Helmholtz problem becomes trivial to solve. Therefore, only one Poisson problem as in Eq. (31) remains to be solved and the computational effort to integrate one time step a priori becomes smaller than for any (semi-)implicit method. Further, the termination threshold for this equation changes correspondingly from Eq. (38) to $\epsilon_A = \|\mathbf{J}^{-1}\|\,\|\mathbf{D}\|\epsilon_H$ where $\|\mathbf{J}^{-1}\|$ is of order one.

The specifications of the numerical setup (i.e. norms, grid orientation, Reynolds number and iteration thresholds) are the same as in the beginning of Section 6. In this section, the multigrid preconditioner for the Poisson problems employs a $V(2,2)$ cycle. The extents of the physical domain are $L_1 \times L_2 \times L_3 = 2\pi/1.12 \times 2 \times 2\pi/2.1$ which is discretized with the d3 scheme on $N_1 \times N_2 \times N_3 = 128 \times 129 \times 128$ grid points. The grid is non-uniform in the wall-normal direction according to Eq. (49) with different values for $\xi = 0, 1, 10, 100$. We investigate the solver performance at three different Reynolds numbers $Re_{\text{lam}} = 50,\ 500,\ 5000$ (based on the maximum velocity of the corresponding laminar Poiseuille flow). The initial conditions

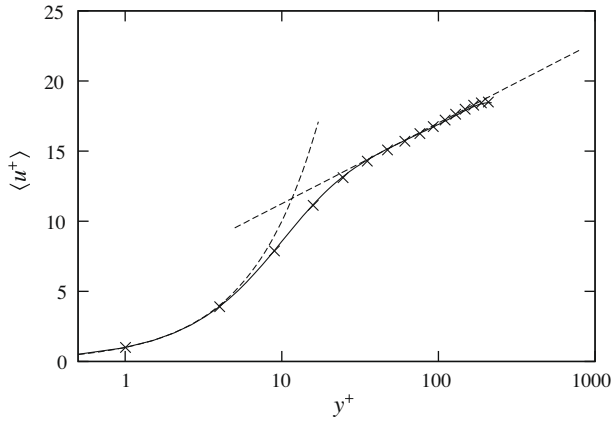

Fig. 14. Mean velocity profile: (————) present simulation, (---) wall laws, × Schlatter [37].

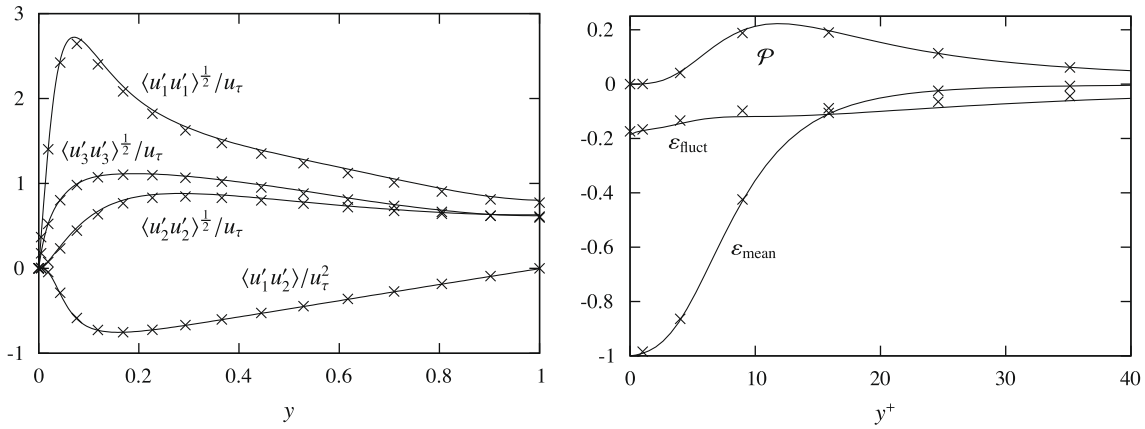

Fig. 15. Reynolds stresses (left) and energy budget normalized by $(Re\ u_\tau^4)$ (right): (————) present simulation, × reference.

are taken from precursor simulations of turbulent channel flow which were conducted for each ξ separately at $Re_{\text{lam}} = 5000$. We integrate the flow with fixed bulk Reynolds number $Re_b = (2/3)Re_{\text{lam}}$ over two time units with the termination threshold $\epsilon_H = 10^{-8}$. The time step size is set to $\Delta t = 0.5\Delta t_{\max}$ where $\Delta t_{\max}$ is the marginally stable time step size according the stability regions of RK3 and CN-RK3, Eq. (4). The simulations are performed on a Cray XT5 supercomputer using $P = 8$ CPU cores. We would like to emphasize that we focus here only on the preconditioner of the pressure iteration which is independent of the parallelization.

The results of this study are listed in Table 6. For strongly non-unifrom grids with small ξ and/or small Reynolds numbers Re (leading to large $\|\mathbf{L}\|_\infty = 2\beta/\Delta t$) the average time step sizes $\langle \Delta t \rangle$ for the RK3 scheme become increasingly small whereas the CN-RK3 scheme is not affected. On the other hand, the elapsed times to integrate one time step are smaller as well since only one pressure iteration is required. Additionally, the flow changes only little for small Δt such that only few iterations are required to solve the Poisson problems (we prescribe at least one Poisson iteration). If we look at the averaged elapsed times $\langle T \rangle$ required to integrate on time unit we find that the semi-implicit scheme is only faster for simulations with large β and in particular with strong grid stretching. More specifically, CN-RK3 is more than six times faster in terms of elapsed time with the commutation-based preconditioner ($\xi = 1,\ Re = 500$). On the other hand, the semi-implicit methods are also slower by a factor of three in cases with almost equidistant grids and high Reynolds numbers which leads to small $\beta(\xi = 1, 10,\ Re = 5000)$. In this parameter regime the time step is mostly limited by convection and less by viscosity.

When we compare the preconditioners in the simulations with CN-RK3 time integration we find that the commutation-based preconditioner (30) performs significantly better than the simpler Laplacian preconditioner (28) in terms of elapsed time and iteration counts. The number of pressure iterations for the latter strongly depends on ξ and Re whereas the commutation-based preconditioner demonstrates a much weaker sensitivity with respect to these parameters. For $\xi,\ Re \to 0$ both preconditioners become increasingly inefficient or lead to divergence (not shown here), however, an appropriately chosen relaxation factor ω or a more sophisticated primary solver may avoid that. The dependence of the preconditioner performance on β is further investigated and discussed in Section 8 for equidistant grids.

Table 6
Performance test of the Laplacian and the commutation-based preconditioner for the outer pressure iteration. $\langle T\rangle$ is the average time in seconds to integrate one time unit, $\langle \Delta t\rangle$ is the average time step size, $\langle l^*\rangle$, $\langle j^*\rangle$, $\langle k^*\rangle$ are the average iteration counts per sub-time step and $\langle \rho_A\rangle \equiv \langle \log_{10}(\|\mathbf{r}_A^{l+1}\|/\|\mathbf{r}_A^{l}\|)\rangle$, $\langle \rho_K\rangle \equiv \langle \log_{10}(\|\mathbf{r}_K^{j+1}\|/\|\mathbf{r}_K^{j}\|)\rangle$, $\langle \rho_H\rangle \equiv \langle \log_{10}(\|\mathbf{r}_H^{k+1}\|/\|\mathbf{r}_H^{k}\|)\rangle$ are the average convergence rates of the pressure, Poisson and Helmholtz iteration, respectively.

$\beta/\Delta t$	Method	$\langle T\rangle$ [s]	$\langle \Delta t\rangle$	$\langle l^*\rangle$	$\langle j^*\rangle$	$\langle k^*\rangle$	$\langle \rho_A\rangle$	$\langle \rho_K\rangle$	$\langle \rho_H\rangle$
$4.58\cdot10^{0}$	CN-RK3, Laplace	$5.31\cdot10^{2}$	$1.55\cdot10^{-2}$	2.12	5.73	6.13	−1.95	−1.21	−4.02
($\xi=100$, $Re=5000$)	CN-RK3, commut.	$5.66\cdot10^{2}$	$1.55\cdot10^{-2}$	1.06	6.96	5.07	−4.33	−1.17	−4.06
	RK3	$1.78\cdot10^{2}$	$2.00\cdot10^{-2}$	1	3.94	n/a	$-\infty$	−1.13	n/a
$2.07\cdot10^{1}$	CN-RK3, Laplace	$6.48\cdot10^{2}$	$1.49\cdot10^{-2}$	3.05	5.88	8.47	−1.33	−1.60	−3.04
($\xi=10$, $Re=5000$)	CN-RK3, commut.	$5.73\cdot10^{2}$	$1.49\cdot10^{-2}$	1.00	5.90	6.15	−3.88	−1.50	−3.24
	RK3	$1.55\cdot10^{2}$	$2.02\cdot10^{-2}$	1	3.32	n/a	$-\infty$	−1.32	n/a
$5.67\cdot10^{2}$	CN-RK3, Laplace	$2.41\cdot10^{3}$	$1.40\cdot10^{-2}$	18.5	19.5	34.4	−.198	−1.77	−1.37
($\xi=1$, $Re=5000$)	CN-RK3, commut.	$9.51\cdot10^{2}$	$1.40\cdot10^{-2}$	2.00	8.17	11.6	−2.04	−1.87	−1.78
	RK3	$1.10\cdot10^{3}$	$1.62\cdot10^{-3}$	1	1.30	n/a	$-\infty$	−1.64	n/a
$6.08\cdot10^{3}$	CN-RK3, Laplace	$1.50\cdot10^{4}$	$1.47\cdot10^{-2}$	105	212	124	−.032	−1.07	−.915
($\xi=0$, $Re=5000$)	CN-RK3, commut.	$1.48\cdot10^{3}$	$1.47\cdot10^{-2}$	2.80	15.7	17.3	−1.39	−1.27	−1.26
	RK3	$5.86\cdot10^{3}$	$3.07\cdot10^{-4}$	1	1.33	n/a	$-\infty$	−.694	n/a
$4.58\cdot10^{1}$	CN-RK3, Laplace	$6.01\cdot10^{2}$	$1.74\cdot10^{-2}$	3.53	6.29	9.47	−1.01	−1.51	−2.59
($\xi=100$, $Re=500$)	CN-RK3, commut.	$5.46\cdot10^{2}$	$1.74\cdot10^{-2}$	1.29	6.55	6.75	−2.85	−1.41	−2.79
	RK3	$2.17\cdot10^{2}$	$1.23\cdot10^{-2}$	1	2.62	n/a	$-\infty$	−1.28	n/a
$2.07\cdot10^{2}$	CN-RK3, Laplace	$1.30\cdot10^{3}$	$1.87\cdot10^{-2}$	11.6	12.4	26.6	−.263	−1.63	−1.26
($\xi=10$, $Re=500$)	CN-RK3, commut.	$7.19\cdot10^{2}$	$1.87\cdot10^{-2}$	2.04	7.95	12.0	−1.86	−1.75	−1.56
	RK3	$6.22\cdot10^{2}$	$3.27\cdot10^{-3}$	1	1.70	n/a	$-\infty$	−1.51	n/a
$5.67\cdot10^{3}$	CN-RK3, Laplace	$1.12\cdot10^{4}$	$1.75\cdot10^{-2}$	128	132	178	8.021	−1.90	−.509
($\xi=1$, $Re=500$)	CN-RK3, commut.	$1.58\cdot10^{3}$	$1.75\cdot10^{-2}$	3.20	12.6	33.1	−.983	−1.74	8.628
	RK3	$9.74\cdot10^{3}$	$1.62\cdot10^{-4}$	1	1.00	n/a	$-\infty$	−2.06	n/a
$4.58\cdot10^{2}$	CN-RK3, Laplace	$1.46\cdot10^{3}$	$2.00\cdot10^{-2}$	13.0	14.3	33.3	−.192	−1.76	−.841
($\xi=100$, $Re=50$)	CN-RK3, commut.	$7.17\cdot10^{2}$	$2.00\cdot10^{-2}$	1.84	7.23	15.0	−1.60	−1.58	−1.06
	RK3	$1.19\cdot10^{3}$	$1.38\cdot10^{-3}$	1	1.11	n/a	$-\infty$	−1.68	n/a
$2.07\cdot10^{3}$	CN-RK3, Laplace	$5.07\cdot10^{3}$	$2.08\cdot10^{-2}$	55.2	57.2	118	−.036	−1.70	−.383
($\xi=10$, $Re=50$)	CN-RK3, commut.	$1.23\cdot10^{3}$	$2.08\cdot10^{-2}$	2.59	8.77	34.5	−1.06	−1.77	−.499
	RK3	$4.83\cdot10^{3}$	$3.27\cdot10^{-4}$	1	1.02	n/a	$-\infty$	−1.98	n/a

In the case of more equidistant grids with larger ξ and Re, the Laplacian preconditioner is slightly more economic since it requires the solution of only one instead of two Poisson problems. However, the difference is benign because the initial residual for the second Poisson problem of the commutation-based preconditioner is already very small such that the second Poisson problem can be solved at very little cost. We conclude that the commutation-based preconditioner (30) enables an efficient solution of the pressure Eq. (20). It is more robust than the Laplacian preconditioner (28), in particular, if we apply strong grid stretching.

8. Parallel performance

This section assesses the computational performance of the proposed simulation code. In particular, we will focus on the performance on massively parallel computers such as the Cray XT or IBM Blue Gene platforms which offer thousands of processors connected by high-bandwidth networks. Again, we use the numerical setup (i.e. norms, grid orientation, Reynolds number, iteration thresholds) as defined at the beginning of Section 6. The multigrid preconditioner for the Poisson problems employs a $V(3,3)$ cycle in this section.

The scalability of the implementation is tested for a channel flow in a cubical domain with edge lengths $L_1 = L_2 = L_3 = L$. The domain is decomposed into $P = P_1 \times P_2 \times P_3$ sub-domains of equal size where P_i are the numbers of processor sub-domains in each direction. It is discretized spatially with the d3 scheme on $N_1 \times N_2 \times N_3 = 128P_1 \times (128P_2 + 1) \times 128P_3$ grid points which are distributed equidistantly in all spatial directions for simplicity (i.e. $\Delta x = const.$, $\xi = \infty$). We investigate the scalability for different $\beta = 0.371,\ 3.710,\ 37.10,\ 371.0$ by varying $\Delta t/(Re\Delta x^2)$ and fixing the initial grid Péclet number $Re\Delta x\|\mathbf{u}(t=0)\| = 1$ [43]. Note that the cases with $\beta \lessapprox s_{\mathrm{visc}}/2 \approx 2.51/2$ can be integrated in time with the explicit RK3 scheme instead of CN-RK3. All results are averaged over 20 time steps, only the tests for $\beta = 371$ are averaged over five time steps due to higher computational costs.

The initial condition is a superposition of a large number of randomly chosen vortices according to Eq. (50) which are reoriented in all three directions. Their amplitudes are scaled with $u_i \sim k_i^{-1}$ to obtain a kinetic energy spectrum reminiscent of

a turbulent energy cascade. To obtain a notable number of pressure iterations and therefore better statistics we choose $\epsilon_H = 10^{-8}$ as the termination threshold for the Helmholtz problems and initialize the pressure field additionally with a constant before each pressure iterations starts. Note that this measure increases the computational effort artificially and would not be performed in production simulations. The initial divergence residual $\|\mathbf{Du}^0(t=0)\| = \|\mathbf{r}_A^0(t=0)\|$ is about the same throughout all tests such that the residual is reduced by a factor of about 10^6 in each sub-time step throughout all simulations.

All tests are performed on a Cray XT5 massively parallel supercomputer which has a 3D torus network and two AMD "Istanbul" hex-core CPUs per node, i.e. there are $n_{\text{cores,max}} = 12$ cores available on each compute node. The tests in this section employ by default double precision arithmetic and range from about $N_1N_2N_3 \approx 2.1 \cdot 10^6$ grid points on one core up to $29.0 \cdot 10^9$ grid points on $P = 13\,824$ cores involving up to 11 grid levels in the multigrid preconditioner for the Poisson problems. Some results in Section 8.2 are conducted with single precision arithmetic which allows us to use up to $N_1N_2N_3 \approx 1.52 \cdot 10^{11}$ grid points on $P = 21\,504$ cores and 12 multigrid levels. All results are obtained in a production environment, i.e. other jobs on the computer avoid an optimal processor mapping to the network nodes and generate additional network traffic. Further, the largest jobs cannot be mapped perfectly to the compute nodes since the arrangement of the sub-domains will be different from the shape of the network mesh ($10 \times 12 \times 16$). To minimize the communication across the network adapters, we assign by default $n_{\text{cores}} = 8$ cubically arranged sub-domains to each node, i.e. four blocks to each CPU.

8.1. Weak scaling

8.1.1. Design and objectives of the test

Because the approximation error of $\widetilde{\mathbf{A}}$ with respect to $\mathbf{A}$ is to first order a function of β (and vanishes for $\beta = 0$) for any of the preconditioners in Section 5.2.1, we expect that the conditioning of $\widetilde{\mathbf{A}}^{-1}\mathbf{A}$ and thus the spectral radius and the convergence rate of the pressure iteration should remain about the same for all test cases for a given β. Similarly, the eigenvalue distribution of the Helmholtz matrices $\mathbf{H}$ depends to first order only on β, such that also the BiCGstab solver for the Helmholtz problems should converge at about the same rate in all test cases with identical β [13]. These relations between β and the convergence rates set the solver performance but do not influence the scalability of the iterative solution algorithm.

In addition to the scalability of the algorithm itself, we require also weak scalability of the implementation on a massively parallel computer with torus network. This means that the computational effort per core and the elapsed time for solving the problem should *ideally* remain the same if the problem is distributed to an increasing number of cores and if the number of unknowns is increased simultaneously (i.e. the number of grid points per core remains constant, $N^3/P = const.$). However, we know from the complexity analysis in Section 3.2 that also a weak dependence on the number of cores P cannot be fully avoided on a torus network due to multigrid.

8.1.2. Results

For performance and simplicity reasons, we decompose the domain into $P = P^{1/3} \times P^{1/3} \times P^{1/3}$ cubical sub-domains in this section. The average iteration counts $\langle l^* \rangle$, $\langle j^* \rangle$, $\langle k^* \rangle$ per sub-time step and convergence rates $\langle \log_{10}(\|\mathbf{r}_A^{l+1}\|/\|\mathbf{r}_A^{l}\|) \rangle$, $\langle \log_{10}(\|\mathbf{r}_K^{j+1}\|/\|\mathbf{r}_K^{j}\|) \rangle$, $\langle \log_{10}(\|\mathbf{r}_H^{k+1}\|/\|\mathbf{r}_H^{k}\|) \rangle$ for the pressure, Poisson and Helmholtz iterations, respectively, are displayed in Fig. 16. As expected, the numbers of iterations as well as the convergence rates remain almost constant for all test cases and depend only on β, which confirms the complexity of the algorithm and therefore its weak scalability. We would like to point out that the convergence rate of the BiCGstab/multigrid Poisson solver is limited by design because the discretization of the Poisson problem (d3 scheme) differs from the fine-grid discretization in the multigrid preconditioner (d1 scheme). For the $V(3,3)$ cycle and the d3 scheme we find that the convergence rate is already close to this limit because an increase of the number of smoothing cycles does not increase the convergence rate significantly any further. This becomes even more evident if the d1 discretization scheme is used instead of the d3 scheme: the convergence rate is approximately two times higher than before and not limited either for larger numbers of smoothing cycles. These convergence rates compare well with other multigrid-preconditioned Krylov subspace solvers, e.g. [1]. Nevertheless, the average convergence rate with the d3 scheme is still sufficiently high for our purposes ($\langle \log_{10}(\|\mathbf{r}_K^{j+1}\|/\|\mathbf{r}_K^{j}\|) \rangle \approx -1.6, \ldots, -1.2$). Additionally, the results demonstrate that the geometric multigrid approach is well suited for such problems.

Because the algorithm scales perfectly and the average numbers of iterations are about the same in all tests with identical β, the overall elapsed time $\langle T \rangle$ to integrate one full time step should ideally remain approximately constant as well for increasing processor counts P. But, as stated before, there actually is a dependence on the number of cores as depicted in Fig. 17. The total elapsed time for pure arithmetic operations is about the same for all tests with identical β such that the timing for pure communication must be responsible for the increase with growing processor counts. From a detailed performance analysis of the case $P = 4^3$ we find that communication takes about 8% of the total execution time. Further testing reveals that the increase of the communication time is mostly caused by the fine-grid ghost cell updates. It does not grow for an ideal processor mapping but for the worst processor mapping it increases with $P^{1/3}$, cf. Section 3.2. The global reductions in the BiCGstab solver as well as the coarse-grid problems of the multigrid preconditioner for the Poisson problems still play a secondary role. They may become visible for much larger processor counts since such operations scale with $P^{1/3}$, independent of the processor mapping (but with a smaller multiplier than the worst-case ghost cell updates). Anyway, the results confirm the complexity of the implementation as expected from Section 3.2.

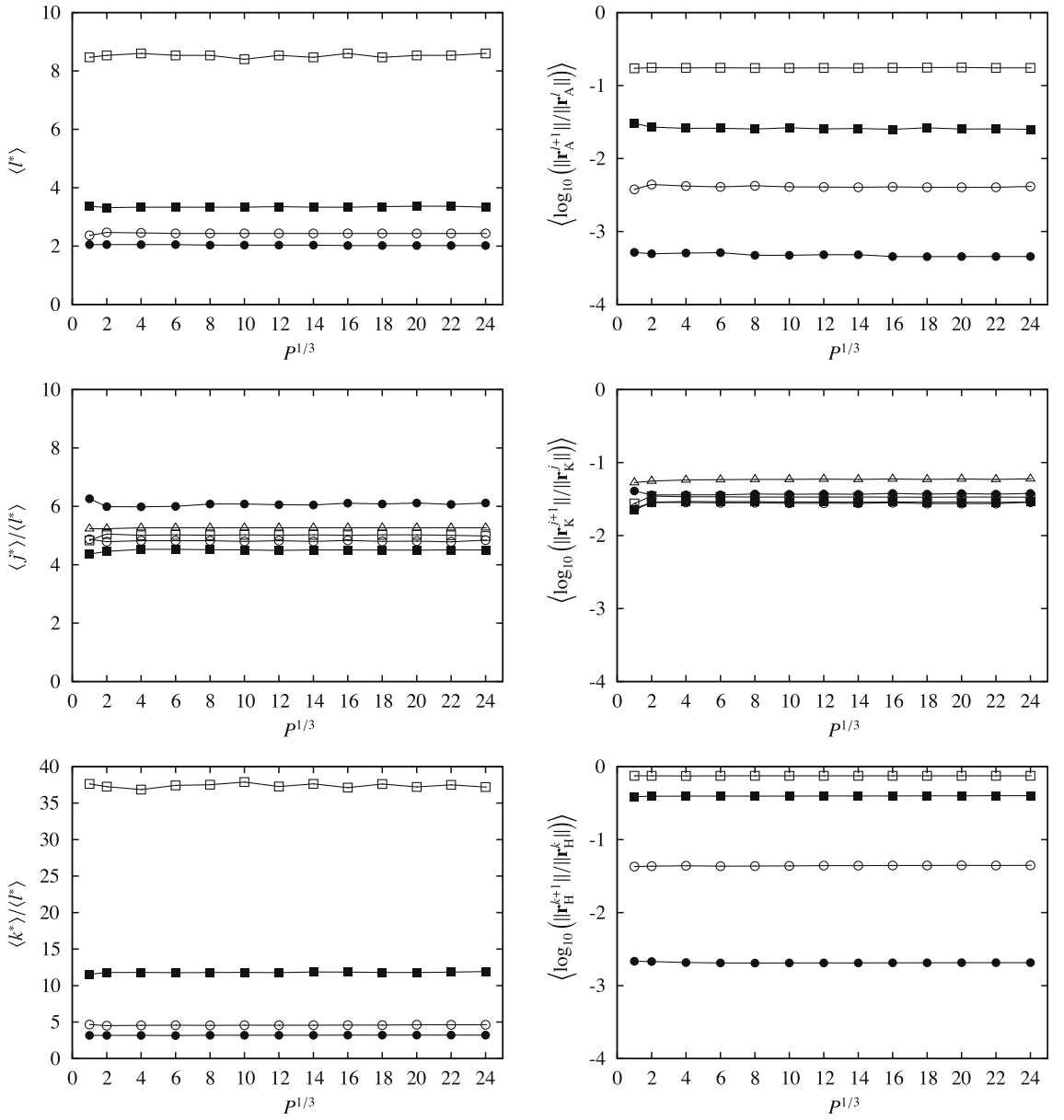

Fig. 16. Weak scalability of the algorithm for CN-RK3 time integration and different β. Left column: average iteration counts per sub-time step (Poisson and Helmholtz problems normalized by $\langle I^* \rangle$), right column: average convergence rates. From top: pressure iteration, Poisson problems, Helmholtz problems. (□) $\beta = 371$, (■) $\beta = 37.1$, (○) $\beta = 3.71$, (●) $\beta = 0.371$, (△) $\beta = 0.371$ and explicit RK3 time integration.

At this point, we would like to comment briefly on the strong increase of the elapsed time from $P_i = P = 1$ to $P_i = 2$ and $P_i = 2$ to $P_i > 2$ in Fig. 17. The case $P = 1$ benefits significantly from a higher memory bandwidth, whereas all other cases use four cores per CPU instead of one. Further, the case $P = 2^3$ is performed on a single compute node where the MPI communication between the processes is much faster than in all larger tests where the network comes into play. By decreasing the number of utilized cores on each node we can reduce the network traffic significantly. Additionally, the effective memory bandwith per core increases, such that the overall execution time benefits. This is demonstrated in Fig. 18 for $\beta = 3.71$ and $n_{\text{cores}} = 2, 4, 8, 12$ sub-domains per node. However, we also find that reducing n_{cores} does not improve the weak scalability.

Obviously, the convergence rate of the pressure iteration decreases with growing β such that all iteration counts increase correspondingly, including the Poisson and Helmholtz problems (cf. Fig. 16). The increase suggests that the convergence rate behaves like $\langle \log_{10}(\|\mathbf{r}_A^{l+1}\| / \|\mathbf{r}_A^l\|) \rangle \approx 0.85 \log_{10}(\beta/5820)$. Since the convergence rate does not vary significantly throughout the iterations and time steps, we can assume that the spectral radius of the pressure iteration is roughly $(\beta/5820)^{0.85} \lessapprox \|\mathbf{I} - \omega \widetilde{\mathbf{A}}^{-1} \mathbf{A}\|$ for $\omega = 1$. The spectral radius and therefore the convergence rate can probably be improved by choosing a different

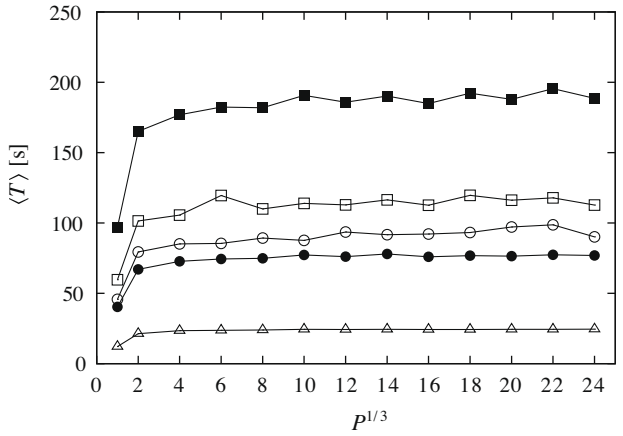

Fig. 17. Weak scaling: average elapsed times per full time step for CN-RK3 time integration (in seconds). (□) $\beta = 371$ ($\langle T\rangle/10$), (■) $\beta = 37.1$, (○) $\beta = 3.71$, (●) $\beta = 0.371$, (△) $\beta = 0.371$ and RK3 time integration.

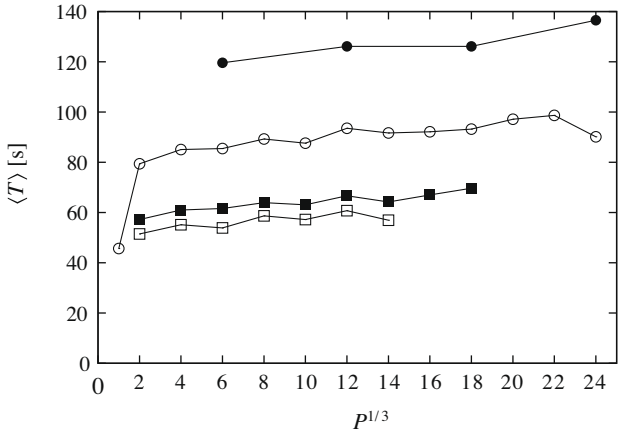

Fig. 18. Weak scaling: average elapsed times per full time step (in seconds). (□) $n_{\mathrm{cores},m}/n_{\mathrm{cores}} = 6$, (■) $n_{\mathrm{cores,max}}/n_{\mathrm{cores}} = 3$, (○) $n_{\mathrm{cores,max}}/n_{\mathrm{cores}} = 3/2$, (●) $n_{\mathrm{cores,max}}/n_{\mathrm{cores}} = 1$.

relaxation factor ω or by using a more sophisticated primary solver such as BiCGstab or GMRES which make a relaxation factor obsolete. Elman et al. [10], for instance, employed a GMRES-type solver for steady-state problems.

Similarly to the pressure iteration, the convergence rate of the Helmholtz problems decreases with increasing β because the character of the Helmholtz problems tends to be more Poisson-like (connected with an increasingly poor conditioning of **H**). It becomes evident that for larger β an appropriate preconditioning (e.g. multigrid) is necessary to reduce the iteration counts to levels which are comparable to those of the Poisson problems. The convergence rate of the Poisson problems is already independent of β and their total iteration count increases only with the iteration count of the pressure iteration. We conclude that preconditioning of the Helmholtz problems and employing a more sophisticated primary solver for the pressure problem may enhance the performance for large β significantly.

Since cases with sufficiently small $\beta \lessapprox s_{\mathrm{visc}}/2 \approx 2.51/2$ do not require (semi-)implicit time integration, we compare the iteration counts and timings between semi-implicit CN-RK3 time integration and purely explicit RK3 time integration for $\beta = 0.371$ (Figs. 16 and 17). As explained in Section 7 we find again that the computational effort to integrate one time step is a priori smaller than for any (semi-)implicit method.

8.2. Strong scaling

In contrast to the weak scaling test, we maintain the problem size $N_1N_2N_3$ constant to test the strong scaling abilities, such that the sub-domains become smaller with increasing numbers of cores P, beginning with $P = P_{\mathrm{ref}}$. Ideally, the elapsed times per full time step T decrease with the number of cores, i.e. $T \sim P^{-1}$ which cannot be expected from the implementation on our test computer (cf. Section 3.2). The scaling tests start with $P_{\mathrm{ref}} = P_{\mathrm{ref},1} \times P_{\mathrm{ref},2} \times P_{\mathrm{ref},3}$ cores where $P_{\mathrm{ref},1} = P_{\mathrm{ref},2} = P_{\mathrm{ref},3} = P_{\mathrm{ref}}^{1/3}$. We increase the number of cores by increasing $P_1,\ P_2,\ P_3$ subsequently by a factor of two. The series ends as

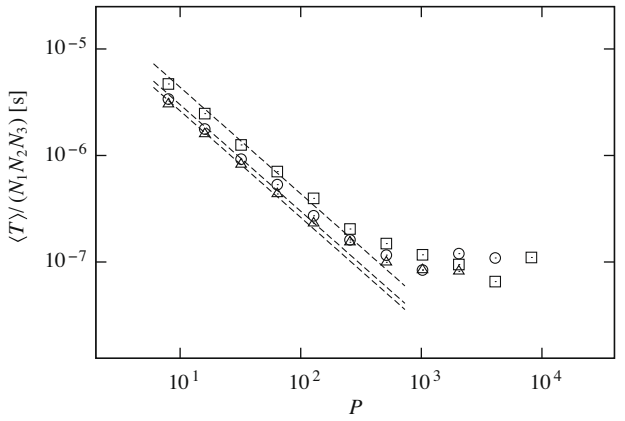

Fig. 19. Strong scaling: average elapsed times per full time step and grid point for $\beta = 3.71$ and $P_{\text{ref}} = 2^3$ (in seconds). (□) $n_{\text{cores,max}}/n_{\text{cores}} = 3/2$, (○) $n_{\text{cores,max}}/n_{\text{cores}} = 3$, (△) $n_{\text{cores,max}}/n_{\text{cores}} = 6$, (---) ideal parallel speed-up.

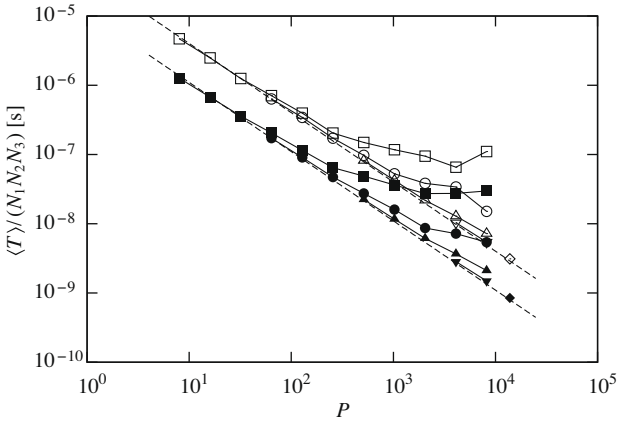

Fig. 20. Strong scaling: average elapsed times per full time step and grid point (in seconds). Empty symbols: $\beta = 3.71$ with CN-RK3 time integration, filled symbols: $\beta = 0.371$ with RK3 time integration. (□) $P_{\text{ref}} = 2^3$, (○) $P_{\text{ref}} = 4^3$, (△) $P_{\text{ref}} = 8^3$, (▽) $P_{\text{ref}} = 16^3$, (◇) $P_{\text{ref}} = 24^3$, (---) ideal parallel speed-up.

soon as the doubling of P exceeds the machine size (note that not all cores of each compute node are employed by default). All timings in this section are in seconds and normalized with the number of grid points $N_1N_2N_3$ to make the results comparable with other approaches and implementations on other computers.

Similarly to Section 8.1, we investigate the strong scalability of the implementation for $P_{\text{ref}} = 2^3$, $\beta = 3.71$ with $n_{\text{cores}} = 2, 4, 8$ sub-domains per node. As demonstrated in Fig. 19 the average elapsed times required to integrate one full time step $\langle T\rangle$, decrease with decreasing n_{cores}. However, the results also reveal the limitations of our approach with respect to its strong scalability. As discussed in Section 3.2, the contributions to the computational complexity of the ghost cell updates and ultimately the coarse-grid communication of the multigrid preconditioner for the Poisson problems come into play for very large processor counts P. In either case, the test demonstrates that the strong scalability cannot be improved any further by a higher effective memory bandwidth per core and minor network load. Moreover, the test indicates that both, memory access and network, are released at about the same rate because the critical number of cores P (at which the elapsed times begin to stagnate) is about the same for the three test series.

Analogously to Section 8.1, we compare semi-implicit (CN-RK3, $\beta = 3.71$) and purely explicit time integration (RK3, $\beta = 0.371$) with respect to the strong scalability of the implementation. The results are depicted in Fig. 20. They demonstrate that the explicit scheme is about 3.7 times faster per full time step than the semi-implicit scheme. However, the parameter β is 10 times larger in the semi-implicit scheme, i.e. the time step size Δt could be larger by the same factor (for fixed Re and Δx) which makes the semi-implicit scheme faster overall. However, this does not account for other aspects such as accuracy issues.

In a next step, we compare our implementation with a state-of-the-art spectral solver (FFT with data transpositions) which was benchmarked by [8] on a computing platform similar to ours. To be able to compare both approaches we need to switch to single precision arithmetic and to enlarge the problem size per core such that the total problem size becomes

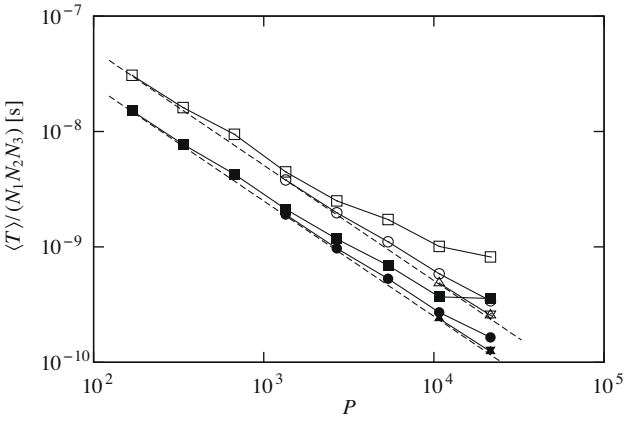

Fig. 21. Strong scaling: average elapsed times per full time step and grid point for $\beta = 0.371$, RK3 time integration, $N_1 N_2 N_3 / P \approx 192^3$ grid points per core and single precision arithmetic (in seconds). Empty symbols: d3 discretization, filled symbols: d1 discretization. (□) $P_{\text{ref}} = 168$, (○) $P_{\text{ref}} = 1\,344$, (△) $P_{\text{ref}} = 10\,752$, (▽) $P_{\text{ref}} = 21\,504$, (---) ideal parallel speed-up.

$N_1 \times N_2 \times N_3 = 192P_1 \times (192P_2 + 1) \times 192P_3$. In this test we employ $n_{\text{cores}} = n_{\text{cores,max}} = 12$ cores per node. Different from the previous tests, we use a base decomposition of $4 \times 6 \times 7$ sub-domains (the largest test problem comprises $32 \times 24 \times 28$ sub-domains). The domain is not cubical anymore since we maintain constant grid spacings $\Delta x = const.$ throughout the test. We use $\epsilon_H = 10^{-6}$ as termination criterion in this test because $\epsilon_H = 10^{-8}$ is too small for single precision arithmetic. Additionally to the d3 discretization scheme we also perform test simulations with the d1 scheme which is computationally cheaper than the d3 scheme. In particular, the convergence rate of the multigrid-preconditioned Poisson solver is larger.

The results are depicted in Fig. 21 for $\beta = 0.371$ and explicit RK3 time integration which demonstrates again good scalability. The smallest average elapsed time per sub-time step and gridpoints per core (i.e. $\langle T \rangle P / (N_1 N_2 N_3)$) is about $\gtrapprox 1.72 \cdot 10^{-6}$ s for the d3 discretization scheme and $\gtrapprox 8.45 \cdot 10^{-7}$ s for the d1 scheme. The spectral method reaches $\gtrapprox 1.2 \cdot 10^{-6}$ s on a Cray XT4 and $\gtrapprox 4.7 \cdot 10^{-6}$ s on a IBM BG/L. Note that these numbers are derived at the maximum core load $N_1 N_2 N_3 / P$ where communication is minimal, such that the processor speeds play a major role. Obviously, our method is competitive at this degree of parallelization and problem size. This has been demonstrated before for much smaller problems, e.g. by Hess and Joppich [18], but not for higher-order discretizations and not for the present problem sizes.

Generally, the elapsed time starts to stagnate at a load of about $N_1 N_2 N_3 / P \lessapprox 10^4$ grid points per core. At this point the network is clearly the limiting factor and not the CPU speed such that the results become more sensitive to network traffic variations due to concurrently running jobs.

9. Concluding remarks

We presented an efficient solver for three-dimensional time-dependent incompressible viscous flows which uses high-order finite difference discretizations in time and space and is optimized for massively parallel supercomputers with a torus network. As laid out in Section 3, a high-fidelity simulation of very large flow problems performs best on such machines with a static data decomposition in all three coordinate directions. This limits our choice for a spatial discretization scheme to local schemes. In the present work, we selected a set of high-order finite difference schemes on staggered grids. They yield stable and accurate results at moderate computational cost and lead to a relatively low amount of communication between the processors. Our choice of a semi-implicit time integration scheme (Crank–Nicolson with a third-order Runge–Kutta scheme) eliminates the restrictive viscous time step limitation. A purely explicit Runge–Kutta time integration has also been implemented which may be more efficient for certain flow configurations (cf. Sections 7 and 8).

For the solution of the discretized equations, we apply a cascade of iterative solution methods. These solvers are well suited for a massively parallel implementation and allow us to meet the desired high accuracy levels at relatively low cost. Depending on the character of the problem (sparsity, conditioning, eigenvalue spectrum, etc.) we use different iterative schemes: a preconditioned Richardson iteration for the outer pressure problem, the Krylov subspace method BiCGstab for the Helmholtz problems and a multigrid-preconditioned BiCGstab scheme for the Poisson problems in the pressure preconditioner.

In contrast to direct solvers, iterative methods allow us to terminate the iteration at a given level of accuracy. We use this property to reduce the computational cost especially for those sub-problems which do not need high accuracy (e.g. preconditioner problems). The relations between the termination criteria for the cascaded iterative solvers were derived in Sections 5.4 and 5.5 and shown to be effective in Section 6.2.

An appropriate choice for the preconditioner of the pressure problem is very critical for the overall efficiency of the simulation code. We chose the commutation-based preconditioner by Elman et al. [10,11] which apparently has not been used

before for time-dependent incompressible flows. We found that this preconditioner performs better than the simpler Laplace preconditioner of Brüger et al. [5]. All of the mentioned preconditioners have in common that the pressure iteration convergences only if $\beta \sim \Delta t/(Re\Delta x^2)$ is sufficiently small for a given relaxation factor ω. This restriction may be avoided by using more sophisticated solvers (e.g. Krylov subspace methods) instead of the Richardson iteration. In either case, a semi-implicit time integration scheme is only efficient if it leads to a faster simulation code compared to a purely explicit time integration. We demonstrated this for the commutation-based preconditioner for a turbulent channel flow.

We validated the implementation carefully by running different test cases. This included a validation of the convergence orders of the temporal and spatial discretization schemes as well as simulations of eigenmodes and of transitional and turbulent channel flows.

The parallel performance tests indicate no apparent limit for the (weak) scalability of the present implementation. Simulations of pseudo-turbulent flow with up to $1.52 \cdot 10^{11}$ grid points were carried out on up to 21 504 CPU cores at an aggregate computational performance level of about 20 Tflop/s. These tests were limited by the size of available computing platform and not by the scalability of the simulation code.

Acknowledgment

This work has been supported through the ETH research grant TH-23/05-2. Computational resources were provided by the Swiss National Supercomputing Centre (CSCS).

References

[1] M. Adams, M. Brezina, J. Hu, R. Tuminaro, Parallel multigrid smoothing: polynomial versus Gauss–Seidel, J. Comput. Phys. 188 (2003) 593–610.
[2] S.W. Armfield, Finite difference solutions of the Navier–Stokes equations on staggered and non-staggered grids, Comput. Fluids 20 (1) (1991) 1–17.
[3] G.A. Blaisdell, N.N. Mansour, W.C. Reynolds, Numerical simulation of compressible homogeneous turbulence, Technical Report Rep. TF-50, Department of Mechanical Engineering, Stanford University, 1991.
[4] A. Brandt, N. Dinar, Multi-grid solutions to elliptic flow problems, in: S. Parter (Ed.), Numerical Methods for Partial Differential Equations, Academic Press, New York, 1979, pp. 53–147.
[5] A. Brüger, B. Gustafsson, P. Lötstedt, J. Nilsson, High order accurate solution of the incompressible Navier–Stokes equations, J. Comput. Phys. 203 (2005) 49–71.
[6] K. Chen, Matrix Preconditioning Techniques and Applications, Cambridge University Press, Cambridge, 2005.
[7] H. Choi, P. Moin, Effects of the computational time step on numerical solutions of turbulent flow, J. Comput. Phys. 113 (1994) 1–4.
[8] D.A. Donzis, P.K. Yeung, D. Pekurovsky, Turbulence simulations on $\mathcal{O}10^4$ processors, in: Proceedings of the TeraGrid 08 Conference, 2008.
[9] T.M. Eidson, G. Erlebacher, Implementation of a fully balanced periodic tridiagonal solver on a parallel distributed memory architecture, Concurrency Comput.: Pract. Exp. 7 (4) (1995) 273–302.
[10] H. Elman, V.E. Howle, J. Shadid, R. Shuttleworth, R. Tuminaro, A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier–Stokes equations, J. Comput. Phys. 227 (3) (2008) 1790–1808.
[11] H.C. Elman, Preconditioning for the steady-state Navier–Stokes equations with low viscosity, SIAM J. Sci. Comput. 20 (1999) 1299–1316.
[12] T. Fujimoto, R.R. Ranade, Two characterization of inverse-positive matrices: the Hawkins–Simon condition and the Le Chatelier–Braun principle, Electron. J. Linear Algebra 11 (2004) 59–65.
[13] A. Greenbaum, Iterative Methods for Solving Linear Systems, SIAM, Philadelphia, 1997.
[14] W. Hackbusch, Multi-Grid Methods and Applications, Springer, Berlin, 1985.
[15] R. Henniger, L. Kleiser, Simulation of gravity-driven flows using an iterative high-order accurate Navier–Stokes solver, in: Direct and Large-Eddy Simulations VII, Trieste, 2008.
[16] R. Henniger, L. Kleiser, E. Meiburg, Direct numerical simulation of a model estuary, in: Sixth Int. Symp. on Turb. Shear Flow Phenom., Seoul, 2009.
[17] V.E. Henson, U.M. Yang, BoomerAMG: a parallel algebraic multigrid solver and preconditioner, Appl. Numer. Math. 41 (2000) 155–177.
[18] R. Hess, W. Joppich, A comparison of parallel multigrid and a fast Fourier transform algorithm for the solution of the Helmholtz equation in numerical weather prediction, Parallel Comput. 22 (1997) 1503–1512.
[19] S. Hoyas, J. Jimenez, Scaling of the velocity fluctuations in turbulent channels up to $Re_\tau = 2003$, Phys. Fluids 18 (2006) 011702.
[20] J. Jeong, F. Hussain, On the identification of a vortex, J. Fluid Mech. 285 (1994) 69–94.
[21] G.E. Karniadakis, M. Israeli, S.A. Orszag, High-order splitting methods for incompressible Navier–Stokes equations, J. Comput. Phys. 97 (1991) 414–443.
[22] L. Kleiser, T.A. Zang, Numerical simulation of transition in wall-bounded shear flows, Annu. Rev. Fluid Mech. 23 (1991) 495–537.
[23] S. Le Borne, Hierarchical matrix preconditioners for the Oseen equations, Comput. Vis. Sci. 11 (3) (2006) 147–157.
[24] S. Le Borne, Block computation and representation of a sparse nullspace basis of a rectangular matrix, Linear Algebra Appl. 428 (11–12) (2008) 2455–2467.
[25] S.K. Lele, Compact finite difference schemes with spectral-like resolution, J. Comput. Phys. 103 (1992) 16–42.
[26] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (4) (1994) 1019–1044.
[27] Y. Li, Wavenumber-extended high-order upwind-biased finite-difference schemes for convective scalar transport, J. Comput. Phys. 133 (1997) 235–255.
[28] A. Lundbladh, D.S. Henningson, A.V. Johansson, An efficient spectral integration method for the solution of the Navier–Stokes equations, Technical Report FFA TN 1992-28, Aeronautical Research Institute of Sweden, FFA, 1992.
[29] L.R. Matheson, E. Tarjan, Analysis of multigrid algorithms on massively parallel computers: architectural implications, J. Parallel Distrib. Comput. 33 (1996) 33–43.
[30] N. Mattor, T.J. Williams, D.W. Hewett, Algorithm for solving tridiagonal matrix problems in parallel, Parallel Comput. 21 (1995) 1769–1782.
[31] P. Moin, J. Kim, On the numerical solution of time-dependent viscous incompressible fluid flows involving solid boundaries, J. Comput. Phys. 35 (1980) 381–392.
[32] P. Moin, K. Mahesh, Direct numerical simulation: a tool in turbulence research, Annu. Rev. Fluid Mech. 30 (1998) 539–578.
[33] K. Moriya, T. Nodera, Breakdown-free ML(k)BiCGStab algorithm for non-hermitian linear systems, in: O. Gervasi et al. (Eds.), Computational Science and its Applications, Lecture Notes in Computer Science, vol. 3483, Springer, 2005, pp. 978–988.
[34] S.V. Patankar, D.B. Spalding, A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows, Int. J. Heat Mass Transfer 15 (10) (1972) 1787–1806.
[35] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 1–39.
[36] A. Povitsky, Parallelization of pipelined algorithms for sets of linear banded systems, J. Parallel Distrib. Comput. 59 (1999) 68–97.

[37] P. Schlatter, Large-eddy simulation of transition and turbulence in wall-bounded shear flow. PhD thesis, ETH Zürich, Zürich, Switzerland, 2005. Diss. ETH No. 16000.
[38] M.P. Simens, J. Jimenez, S. Hoyas, Y. Mizuno, A high-resolution code for turbulent boundary layers, J. Comput. Phys. 228 (2009) 4218–4231.
[39] P.R. Spalart, R.D. Moser, M.M. Rogers, Spectral methods for the Navier–Stokes equations with one infinite and two periodic directions, J. Comput. Phys. 96 (1991) 297–324.
[40] P.N. Swarztrauber, S.W. Hammond, A comparison of optimal FFTs on torus and hypercube multicomputers, Parallel Comput. 27 (2001) 847–859.
[41] S. Turek, A comparative study of time-stepping techniques for the incompressible Navier–Stokes equations, Int. J. Numer. Methods Fluids 22 (10) (1996) 987–1011.
[42] H.A. van der, Vorst. BiCGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Comput. 13 (1992) 631–644.
[43] J. Wesseling, Principles of Computational Fluid Dynamics, Springer, Berlin, 2001.
[44] G. Wittum, Multi-grid methods for Stokes and Navier–Stokes equations—transforming smoothers: algorithm and numerical results, Numer. Math. 54 (1989) 543–563.
[45] A.A. Wray, Very low storage time-advancement schemes, Technical report, NASA Ames Research Center, 1986.
[46] J. Xu, Benchmarks on tera-scalable models for DNS of turbulent channel flow, Parallel Comput. 33 (12) (2007) 780–794.
[47] F. Zhang, The Schur Complement and its Applications, Springer, New York, 2005.