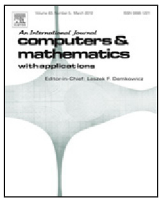
Computers and Mathematics with Applications 76 (2018) 1853–1862

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

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



A FFT-based finite-difference solver for massively-parallel direct numerical simulations of turbulent flows[☆]

Pedro Costa

KTH, Department of Mechanics, SE-100 44 Stockholm, Sweden

A R T I C L E I N F O

Article history:
Received 13 March 2018
Received in revised form 26 June 2018
Accepted 21 July 2018
Available online 3 August 2018

Keywords:
Direct numerical simulations
Turbulent flows
High-performance computing
Fast Poisson solver

A B S T R A C T

We present an efficient solver for massively-parallel direct numerical simulations of incompressible turbulent flows. The method uses a second-order, finite-volume pressure-correction scheme, where the pressure Poisson equation is solved with the method of eigenfunction expansions. This approach allows for very efficient FFT-based solvers in problems with different combinations of homogeneous pressure boundary conditions. Our algorithm explores all combinations of pressure boundary conditions valid for such a solver, in a single, general framework. The method is implemented in a 2D *pencil*-like domain decomposition, which enables efficient massively-parallel simulations. The implementation was validated against different canonical flows, and its computational performance was examined. Excellent strong scaling performance up to 10^4 cores is demonstrated for a domain with 10^9 spatial degrees of freedom, corresponding to a very small wall-clock time/time step. The resulting tool, *CaNS*, has been made freely available and open-source.

1. Introduction

Turbulent flows are ubiquitous in nature and industry, being the most common flow regime for cases which dimensions the human eye can depict. These flows exhibit unsteady, three-dimensional, chaotic and multi-scale dynamics. The Navier–Stokes equations governing its dynamics are highly non-linear, making analytical predictions often difficult. Fortunately, the continuous increase in computer power, together with the progress in development of efficient numerical techniques, resulted in a paradigm change in turbulence research. It is now possible to conduct direct numerical simulations (DNS) which generate data for the full spectrum of scales with billions, or even trillions of spatial degrees of freedom [1].

Pseudo-spectral approaches have been the method of choice for DNS of relatively simple flows, like homogeneous isotropic turbulence [2] or pressure-driven wall-bounded flows [3,4]. In these cases one can benefit from the spectral spatial convergence of the solution, and explore the FFT algorithm for efficient computations. In many situations, however, lower-order discretizations are desirable. Finite-difference methods, for instance, can reproduce several lower-order moments of a flow observable with the same fidelity of a spectral method, and much less computational effort [5,6] (despite requiring more resolution, and consequently higher memory bandwidth per FLOP). Moreover, these methods are in general more versatile in terms of boundary conditions and problem complexity [7]. For instance, in finite-difference algorithms, complex geometries can be easily and efficiently implemented through immersed-boundary methods [8,9], without facing problems due to Gibbs phenomenon.

Indeed, many fundamental insights on the dynamics of turbulent flows have been revealed with standard second-order finite-difference algorithms embedded in a pressure-correction scheme. Few of many examples are the turbulent channel

[☆] The numerical tool is freely available in github.com/p-costa/CaNS.
E-mail address: pedrosc@mech.kth.se.

https://doi.org/10.1016/j.camwa.2018.07.034
0898-1221/© 2018 Elsevier Ltd. All rights reserved.

flow with rough walls [10], flow through porous media [11], interface-resolved simulations of particle-laden flows [12], and turbulent Rayleigh–Bénard convection [13]. This class of methods allows for fast computations, and is often used e.g. when a dense disposition of immersed boundaries reduces anyway the overall accuracy of the method to second-order.

The Poisson equation for the correction pressure is often the most demanding part of the Navier–Stokes solver. Consider the second-order finite-difference discretization of the Laplacian operator. In many cases, iterative solvers (e.g. multigrid methods [14]) are used to solve the resulting system. These methods exhibit excellent scaling properties and are versatile when it comes to the type of boundary conditions that can be implemented. However, much more efficient direct solvers can be used for several combinations of homogeneous pressure boundary conditions. The method of eigenfunction expansions [15–17] is an example. This approach has been implemented in the 1980s in the well-known `FISHPAK` library [18] and explores the fact that the number of diagonals in the coefficient matrix can be reduced by solving an eigenvalue problem with Fourier-based expansions. When compared to multigrid methods, FFT-based direct solvers for the second-order finite-difference Poisson equation can be $O(10)$ times faster [19], and by construction satisfy the solution to machine precision.

Consider, for instance, a 3D problem with at least two periodic boundary conditions. Using the method of eigenfunction expansions, one can apply a discrete Fourier transform (DFT) operator in two directions, and solve a resulting tridiagonal system with Gauss elimination. The inherent challenge for an efficient parallelization is that the FFT algorithm requires all the points in one direction. Consequently, a distributed-memory parallelization in more than one direction is not straightforward. This difficulty has restricted the progress in scaling these methods to a very high number of cores (e.g. more than $O(10^3)$).

There has been a change in trend since highly-scalable libraries for two-dimensional *pencil-like* domain decomposition [20] started to appear. Several recent examples of numerical implementations using this direct method, combined with a 2D domain decomposition, achieved unprecedented performances in problem sizes with $O(10^9)$ grid points. Examples are turbulent Taylor–Couette flows [21], interface-resolved simulations of bubbly flows in homogeneous isotropic turbulence [22], and interface-resolved simulations of turbulent wall-bounded suspensions [23].

Recently, van der Poel et al. [24] published a numerical algorithm[1] for wall-bounded turbulence that showed very good performance both in terms of scaling, up to 64 000 cores, and in terms of actual wall-clock time. In their work and the studies mentioned in the previous paragraph, discrete Fourier transforms (DFT) could be applied to solve the second-order finite-difference Poisson equation, as there were at least two periodic directions. However, other types of FFT-based expansions can be used for several combinations of homogeneous pressure boundary conditions [7,17,25]. To the best of our knowledge, there is no general implementation for massively-parallel DNS that features the wide range of boundary conditions that can be covered with a second-order, FFT-based solver. This is the main motivation of the present work.

We present a numerical method for fast, massively-parallel numerical simulations of turbulent flows. The corresponding code, *CaNS* (Canonical Navier–Stokes), benefits from the efficiency of FFT-based codes for the finite-difference discretization of the pressure Poisson equation, and allows for simulating a wide range of canonical flows.

The manuscript is organized as follows. In Section 2 we describe the method and provide some mathematical background on the Poisson solver. Section 3 describes details on the implementation of the numerical algorithm for massively-parallel numerical simulations of turbulent flows. Section 4 presents a validation of the code for distinct flows, and accesses its computational performance in thousands of cores. Finally, in Section 5 we conclude and discuss future perspectives.

2. Numerical method

The numerical algorithm solves the Navier–Stokes equations for an incompressible, Newtonian fluid with unit density, and kinematic viscosity ν,

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

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u}, \tag{1b}$$

where $\mathbf{u}$ and p are the fluid velocity vector and pressure, respectively.

These equations are solved in a structured Cartesian grid, uniformly-spaced in two directions. Standard second-order finite-differences are used for spatial discretization with a staggered (marker and cell) disposition of grid points. The equations above are coupled through a pressure-correction method [26], and integrated in time with a low-storage, three-step Runge–Kutta scheme (RK3) [27]. The advancement at each substep k is presented below in semi-discrete notation ($k = 1, 2, 3$; $k = 1$ corresponds to a time level n and $k = 3$ to $n + 1$):

$$\mathbf{u}^* = \mathbf{u}^k + \Delta t \left(\alpha_k \mathbf{AD}^k + \beta_k \mathbf{AD}^{k-1} - \gamma_k \nabla p^{k-1/2}\right), \tag{2a}$$

$$\nabla^2 \Phi = \frac{\nabla \cdot \mathbf{u}^*}{\gamma_k \Delta t}, \tag{2b}$$

$$\mathbf{u}^k = \mathbf{u}^* - \gamma_k \Delta t \nabla \Phi, \tag{2c}$$

$$p^{k+1/2} = p^{k-1/2} + \Phi, \tag{2d}$$

1 Freely available in github.com/PhysicsofFluids/AFiD.

where $\mathbf{AD} \equiv -(\mathbf{u}\cdot\nabla)\mathbf{u}+\nu\nabla^2\mathbf{u}$, $\mathbf{u}^*$ is the prediction velocity and Φ the correction pressure. The RK3 coefficients are given by $\alpha_k = \{8/15, 5/12, 3/4\}$, $\beta_k = \{0, -17/60, -5/12\}$ and $\gamma_k = \alpha_k + \beta_k$. This temporal scheme has been proven to be reliable for DNS of turbulent flows, yielding overall second-order temporal and spatial accuracy [28].

A sufficient criterion for a stable temporal integration is given in [27]:

$$\Delta t < \min\left(\frac{1.65\Delta r^2}{\nu}, \frac{\sqrt{3}\Delta r}{\max_{ijk}(|u|+|v|+|w|)}\right), \tag{3}$$

with $\Delta r = \min(\Delta x, \Delta y, \Delta z)$ and Δx_i the grid spacing in direction $x_i \equiv \{x, y, z\}$. The time step restriction due to the viscous effects can be removed with an implicit discretization (e.g. Crank–Nicolson) of the diffusion term. This involves the solution of three Helmholtz equations, one for each component of the prediction velocity. The associated computational overhead can pay off in case of flows at low Reynolds number, but is unnecessary for the inertia-dominated flows of our interest.

Finally, let us note that the boundary conditions for the correction pressure and prediction/final velocity may not be specified independently. For instance, for a prescribed velocity, the pressure boundary condition must be set to homogeneous Neumann (i.e. zero gradient in the direction normal to the boundary), such that the projection step does not alter this condition.

2.1. Poisson equation

Often the solution of the Poisson equation for the correction pressure, Eq. (2b), is the most computation-intensive part of a Navier–Stokes solver. Even so, there are several configurations for which a fast (FFT-based), direct method can be used, even if the unknown is non-periodic. To achieve this, one can explore the method of eigenfunction expansions, as in [17]. This method is applied in two domain directions, x and y, requiring therein a constant grid spacing, and homogeneous boundary conditions. Since this part of the algorithm comprises the most elaborate parallelization steps, we will briefly introduce some mathematical background below.

Consider the following constant-coefficients Poisson equation discretized with second-order central differences at grid cell i, j, k:

$$\begin{aligned}&(\Phi_{i-1,j,k} - 2\Phi_{i,j,k} + \Phi_{i+1,j,k})/\Delta x^2 +\\ &(\Phi_{i,j-1,k} - 2\Phi_{i,j,k} + \Phi_{i,j+1,k})/\Delta y^2 +\\ &(\Phi_{i,j,k-1} - 2\Phi_{i,j,k} + \Phi_{i,j,k+1})/\Delta z^2 = f_{i,j,k};\end{aligned} \tag{4}$$

The method reduces this system of equations with 7 non-zero diagonals to a tridiagonal system, which can be solved very efficiently with Gauss elimination. To achieve this we apply a discrete operator $\mathcal{F}^{x_i}$ to Eq. (4) in two domain directions, that reduces the problem to:

$$(\lambda_i/\Delta x^2 + \lambda_j/\Delta y^2)\hat{\hat{\Phi}}_{i,j,k} + (\hat{\hat{\Phi}}_{i,j,k-1} - 2\hat{\hat{\Phi}}_{i,j,k} + \hat{\hat{\Phi}}_{i,j,k+1})/\Delta z^2 = \hat{\hat{f}}_{i,j,k}, \tag{5}$$

where $\hat{\hat{\square}} \equiv \mathcal{F}^y(\mathcal{F}^x(\square))$ and, although not necessary, we consider Δz constant for simplicity.[2] The eigenvalues λ and operators $\mathcal{F}^{x_i}$ depend on the boundary conditions of the problem, which need to be satisfied by the corresponding inverse (backward) operator, $\mathcal{F}^{-1x_i}$ (e.g., for a periodic boundary condition $\mathcal{F}$ is the Discrete Fourier Transform (DFT)). For several non-periodic combinations of boundary conditions, $\mathcal{F}$ corresponds to well-known discrete transforms (DT) that can be expressed in terms of DFT; see [7,17]. This allows for efficient FFT algorithms with little computational overhead with respect to the periodic case. An operator could in principle be applied a third time, in direction z. However, the tridiagonal system in Eq. (5) is solved more efficiently with Gauss elimination: $O(n)$ operations, contrasting with $O(n\log n)$ required by the FFT algorithm, with n being the number of grid cells in z. Moreover, a non-uniform grid spacing in z is possible with Gauss elimination.

Tables 1 and 2 summarize the transforms pertaining to different boundary conditions. Since the pressure grid cells in the Navier–Stokes solver are staggered, we solely present the transforms whose inverse satisfy homogeneous staggered boundary conditions. Note, however, that non-staggered versions must be used when the temporal integration of the diffusion term in Eq. (2a) is implicit.

Finally, the steps required for solving the Poisson, and associated number of operations are shown below (with N_{x_i} the number of points in direction x_i):

1: compute N_z times the discrete forward transform of f in directions x and y successively, $\hat{\hat{f}} = \mathcal{F}^y(\mathcal{F}^x(f)) \rightarrow \mathcal{O}(N_z(N_yN_x\log N_x + N_xN_y\log N_y))$ operations;

2: solve the resulting N_xN_y tridiagonal systems for the pressure with Gauss elimination – $\mathcal{O}(N_xN_yN_z)$ operations;

3: compute N_z times the discrete backward transform of $\hat{\hat{f}}$ in directions y and x successively, $f = \mathcal{F}^{-1,x}(\mathcal{F}^{-1,y}(\hat{\hat{f}})) \rightarrow \mathcal{O}(N_z(N_xN_y\log N_y + N_yN_x\log N_x))$ operations.

[2] In the numerical tool, the grid spacing in z can be non-uniform.

Table 1
Eigenvalues, and forward ($\mathcal{F}$) and backward ($\mathcal{F}^{-1}$) transforms for different combinations of boundary conditions. P, D and N denote, respectively, periodic, and staggered Dirichlet and Neumann boundary conditions (BC). The eigenvalues (Eq. (5)) are given by $\lambda_q = -4\sin^2(\theta_q)$, $q = 0, 1, \dots, n-1$ [17]; $p = 0, 1, \dots, n/2-1$ and n is the (even) number of grid cells in the one direction. The mathematical expressions for $\mathcal{F}$ are shown in Table 2.

BC	θ_q	$\mathcal{F}$	$\mathcal{F}^{-1}$
P–P	$\begin{cases} \dfrac{(p+1)\pi}{n} & , q = 2p+1 \\ \theta_{p-1} & , q = 2p \neq 0 \\ 0 & , q = 0 \end{cases}$	DFT	$\frac{1}{n}$ IDFT
N–N	$\frac{q\pi}{2n}$	DCT-II	$\frac{1}{2n}$ DCT-III
D–D	$\frac{(q+1)\pi}{2n}$	DST-II	$\frac{1}{2n}$ DST-III
N–D	$\frac{(2q+1)\pi}{4n}$	DCT-IV	$\frac{1}{2n}$ DCT-IV

Table 2
Coefficients $\hat{a}_q$, ($q = 0, 1, \dots, n-1$) for a discrete transform $\hat{a} = \mathcal{F}(a)$ of a sequence of n real numbers $a \equiv \{a_0, a_1, \dots, a_{n-1}\}$ [17]. n is assumed to be even and $l = 0, 1, \dots, n/2-1$.

$\mathcal{F}$	$\hat{a}_q$
DFT	$\begin{cases} \sum_{p=0}^{n-1} a_p & , q = 0 \\ \sum_{p=0}^{n-1} a_p \cos(2\pi pl/n), & q = 2l+1 \neq n-1 \\ \sum_{p=0}^{n-1} a_p \sin(2\pi pl/n), & q = 2l \neq 0 \\ \sum_{p=0}^{n-1} a_p (-1)^p & , q = n-1 \end{cases}$
IDFT	$a_0 + \sum_{p=0}^{n/2-2} (a_{2p+1} \cos(2\pi pq/n) + a_{2p+2} \sin(2\pi pq/n)) + a_{n-1}(-1)^q$
DCT-II	$2\sum_{p=0}^{n-1} a_p \cos(\pi(p+1/2)q/n)$
DCT-III	$a_0 + 2\sum_{p=1}^{n-1} a_p \cos(\pi p(q+1/2)/n)$
DST-II	$2\sum_{p=0}^{n-1} a_p \sin(\pi(p+1/2)(q+1)/n)$
DST-III	$2\sum_{p=0}^{n-2} a_p \sin(\pi(p+1)(q+1/2)/n) + (-1)^q a_{n-1}$
DCT-IV	$2\sum_{p=0}^{n-1} a_p \cos(\pi(p+1/2)(q+1/2)/n)$

3. Implementation

The numerical algorithm is implemented in FORTRAN90/95, with a Message-Passing Interface (MPI) extension for distributed-memory parallelization, combined with a shared-memory parallelization (hybrid MPI-OpenMP). The geometry is divided into several computational subdomains in a *pencil*-like decomposition (Fig. 1). Throughout most steps of the algorithm, $N_p^x \times N_p^y$ pencils are aligned in the z direction.

The 2-cell width of the finite-difference discretization requires communication. Following common practice, we use *halo* cells that store a copy of data pertaining to the boundary of an adjacent subdomain. This requires four pairwise data exchanges (e.g. SEND_RECV) per halo update, and has negligible computational overhead.

Further communication steps are required for computing the DT: the operators are applied successively in directions x and y, requiring all the grid cells for the direction considered. A simple solution would be to change the distribution from 2D to a 1D *slab*-like configuration, decomposing the domain only in z. This requires a single MPI_ALL_TO_ALL operation, but restricts the number of slabs to $N_p^x N_p^y \leq N_z$. Combining this approach with a shared-memory parallelization (e.g. hybrid MPI-OpenMP) relaxes the restriction by a factor $O(10)$ for the present state-of-the-art present hardware (see e.g. top500.org), which can still be constraining for relatively large problem sizes, e.g. with $N_z = O(10^2)$ and very large $N_x N_y$.

This issue can be circumvented by keeping the 2D decomposition, and transposing the data distribution such that it is shared in the direction of interest, as in Fig. 1. This can be done at the cost of a one extra *all-to-all* operation per DT. To achieve this we use the highly-scalable 2DECOMP&FFT library [20]. This library provides a simple interface to transpose the decomposition from x to y-aligned pencils, from y to z, and the reciprocal operations y to x and z to y. The authors of the AFiD code [24] implemented a direct transposition from x-aligned to z-aligned pencils and vice-versa, which was not present in the original 2DECOMP&FFT library and avoids one extra *all-to-all* operation when transposing from x to z. We ported these modifications to our DNS code, although in the present study we still use the original transpose routines from 2DECOMP&FFT.

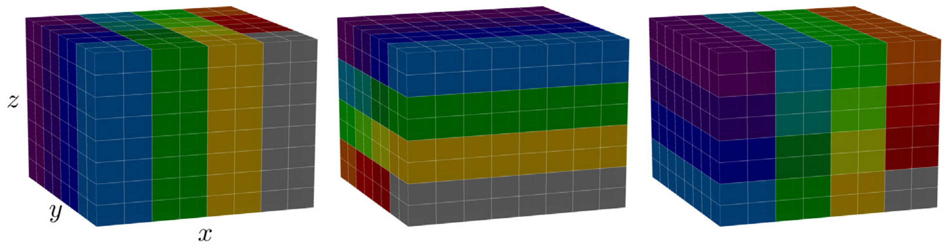


Fig. 1. Illustration of a pencil-like domain decomposition, with $N_x \times N_y \times N_z = 8^3$ grid cells and $N_p^x \times N_p^y = 4^2$ computational subdomains (color-coded). The left-most panel corresponds to the configuration used throughout most steps of the algorithm. Transpositions of the data distribution to x-aligned (middle) and y-aligned (right) pencils are performed in the Poisson solver to compute efficiently the discrete transforms $\mathcal{F}^{x_i}$.

Table 3
Physical and computational parameters for the validation cases. L_{x_i} and N_{x_i} denote the domain size and number of points in direction x_i, respectively. See the text for the scaling parameters used for the Reynolds number, Re.

Case	$L_x \times L_y \times L_z$	$N_x \times N_y \times N_z$	Pressure BC in x,y,z	Re
Lid-driven cavity	$1 \times 1 \times 1$	$128 \times 128 \times 128$	`N-N, N-N, N-N`	1000
Square duct	$10 \times 1 \times 1$	$512 \times 128 \times 128$	`P-P, N-N, N-N`	4410
Plane channel	$6 \times 3 \times 1$	$512 \times 256 \times 144$	`P-P, P-P, N-N`	5640
Taylor–Green vortex	$2\pi \times 2\pi \times 2\pi$	$512 \times 512 \times 512$	`P-P, P-P, P-P`	1600

The discrete transforms are computed with the `FFTW3` library [29]. Apart from its inherent efficiency, the *Guru* interface of the library computes all the discrete transforms presented in Table 2 with the same syntax, just by modifying a parameter corresponding to the transform type[3]. This allowed for a single and efficient parallel implementation of the Poisson solver, as only one data layout for the transposing routines is required. The resulting tridiagonal system of equations is solved with the `LAPACK` library (`DGTSV`) [30]. In case of periodicity in z, the resulting cyclic tridiagonal system is reduced into two tridiagonal problems; see e.g. [31].

The algorithm for solving the Poisson equation is summarized below:

```
1: compute the RHS of the Poisson equation in the $z$-aligned pencil decomposition, and transpose result to $x$-aligned pencil decomposition;
2: compute $N_yN_z$ forward 1D DT in $x$, and transpose result to $y$-aligned pencil decomposition;
3: compute $N_xN_z$ forward 1D DT in $y$, and transpose result to $z$-aligned pencil decomposition;
4: solve $N_xN_y$ linear tridiagonal systems with Gauss elimination, and transpose result to $y$-aligned pencil decomposition;
5: compute $N_xN_z$ backward 1D DT in $y$, and transpose result to $x$-aligned pencil decomposition;
6: compute $N_yN_z$ backward 1D DT in $x$, and transpose result to $z$-aligned pencil decomposition.
```

Finally, parallel I/O is also handled by the `2DECOMP&FFT`, which is based on `MPI-I/O`.

4. Validation and computational performance

4.1. Validation

We validated our implementation against three wall-bounded flows, with different combinations of boundary conditions, as shown in Table 3. Hereafter, u, v and w denote the x-, y- and z-components of the mean velocity, respectively.

4.1.1. Lid-driven cavity flow

We simulated a lid-driven cavity flow in a cubic domain with dimensions $[-h/2, h/2]^3$. Zero-velocity no-slip and no-penetration boundary conditions are prescribed at all the boundaries, except for the top wall, which moves with a velocity $\mathbf{u}(x, y, h/2) = (U_L, 0, 0)$. Other physical and computational parameters are shown in Table 3, where the Reynolds number is defined as $\mathrm{Re} = U_L h/\nu$.

Fig. 2 shows the velocity profiles of the steady state solution at the centerlines $u(0, 0, z)$ and $w(x, 0, 0)$, compared to the data extracted from Ku et al. [32]. The results show excellent agreement with the data.

[3] Note that the result of a *real-to-complex* DFT of n real numbers only requires storage of n real numbers, and therefore can be handled in the same way as the other DT.

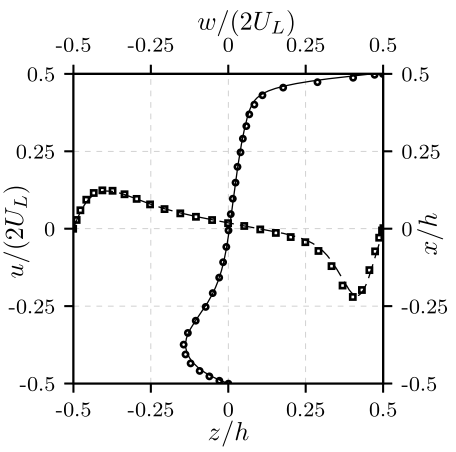


Fig. 2. Normal velocity profiles along the centerlines $u(0, 0, z)$ and $w(x, 0, 0)$ for a lid-driven cubic cavity at Re $=$ 1000. The symbols correspond to DNS data extracted from [32].

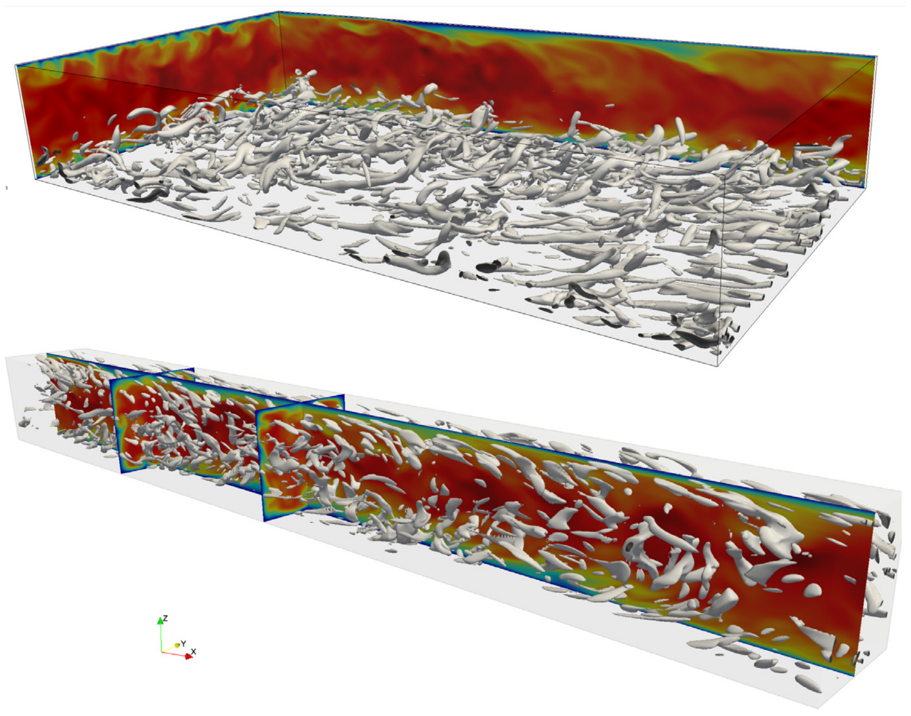


Fig. 3. Visualizations of the simulations of the turbulent channel (top) and duct (bottom). Q-criterion shown with iso-contours of the second invariant of the velocity-gradient tensor, $Q/(U_b/(2h))^2 = 5$ (for the channel only the lower half is shown). The contours pertain to streamwise velocity (red—high, blue—low). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.1.2. Pressure-driven turbulent channel and square duct flow

We now consider two turbulent wall-bounded flows: a plane channel and a square duct. Both flows are periodic in the streamwise (x) direction, with no-slip/no-penetration boundary conditions at the wall-normal directions ($y = \pm h$ and $z = \pm h$ in the case of the duct, and $z = \pm h$, with periodicity in the spanwise direction y, in the channel case). A volume force is added to the discretized momentum equation, to maintain a bulk streamwise velocity $U_b = 1$. The physical and computational parameters are shown in Table 3, where Re $= U_b(2h)/\nu$, and h is the channel/duct half-height.

Fig. 3 shows a 3D visualization for the two flow cases, illustrating what is typically seen for a turbulent flow at low Reynolds number: three-dimensional coherent structures with a relatively small scale-separation with respect to the scales of confinement.

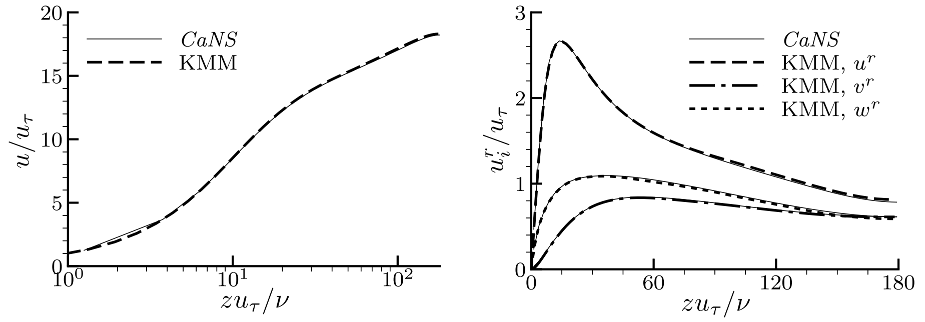


Fig. 4. Left: mean streamwise velocity profile for turbulent channel flow at friction Reynolds number $\mathrm{Re}_\tau = 180$. Right: profiles of root-mean-square velocity u_i^r. Both figures use inner-scaling, i.e. velocity scaled with the wall friction velocity u_τ, and distance with the viscous wall-unit ν/u_τ. The profiles are compared to DNS data from [3] (KMM).

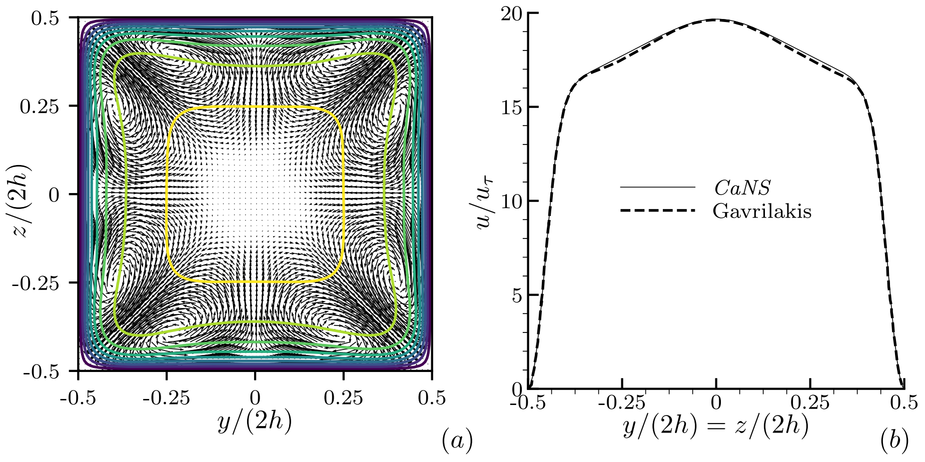


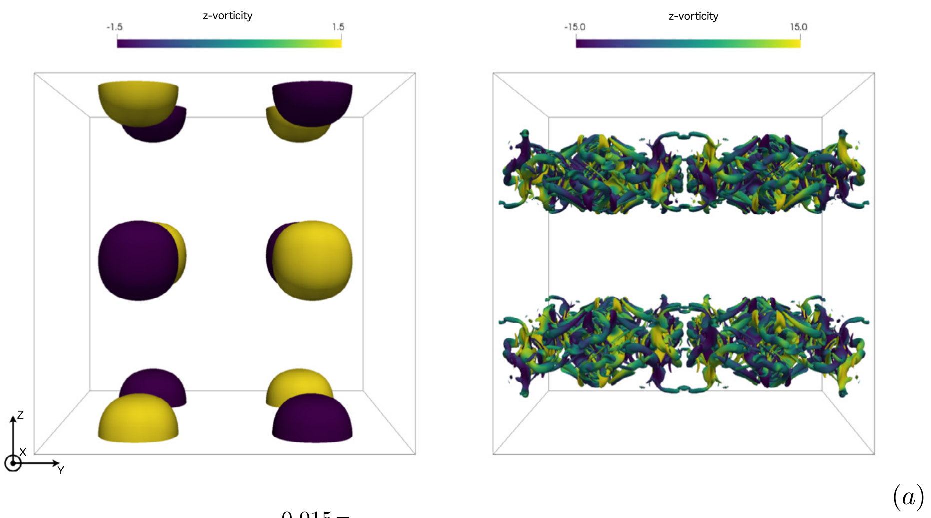
Fig. 5. (a) Mean flow for a square duct. The isolines pertain to streamwise velocity, starting with $2u_\tau$, and evenly-spaced by the same amount. The vectors illustrate in-plane velocity, with a maximum magnitude of $0.02U_b$. (b) Mean streamwise velocity along the duct diagonal, compared to the data extracted from Gavrilakis [4].

The channel flow simulation was initialized with a streamwise vortex pair [33] which effectively triggered transition. Statistics were collected once the mean pressure gradient required to sustain the constant flow rate reached a statistically steady state. The data were ensemble-averaged from 1000 samples, over a period of $1400h/U_b$.

We computed the friction Reynolds number from the time-averaged pressure gradient, yielding $\mathrm{Re}_\tau = u_\tau h/\nu = 180.2$, with $u_\tau \equiv \sqrt{-(\mathrm{d}p_m/\mathrm{d}x)h}$ the wall-friction velocity. This agrees with the correlation given by Pope [34]: $\mathrm{Re}_\tau = 0.09\mathrm{Re}^{0.88} \approx 180$. Fig. 4 compares first and second-order mean flow statistics against data from the seminal paper of Kim et al. [3] at the same friction Reynolds number. Again, the results closely match the reference DNS data.

Now for the turbulent square duct, we used an initial condition from a DNS performed in our group at similar Reynolds number, which allows for a fast transient towards a fully-developed turbulent state. Since the mean flow is two-dimensional, statistical convergence of the results requires many samples. The data were averaged from 1000 samples, over a period of $15000h/U_b$.

Like for the turbulent channel, we compute the friction Reynolds number based on the wall-friction velocity, obtaining a value of $\mathrm{Re}_\tau = 149.1$, consistent with the value of ≈ 150 reported by Gavrilakis [4] for a DNS with the same parameters. Fig. 5(a) quantifies the mean flow. It is well-known that the presence of corners induces a non-zero (secondary) mean flow in the wall-normal directions, in this case with a maximum magnitude of about 2% of the bulk velocity. This value is also consistent with the results reported by Gavrilakis [4]. Panel (b) of Fig. 5 compares the streamwise velocity profile along the diagonal of the duct cross-section, compared to the data extracted from [4]. The results show good agreement.



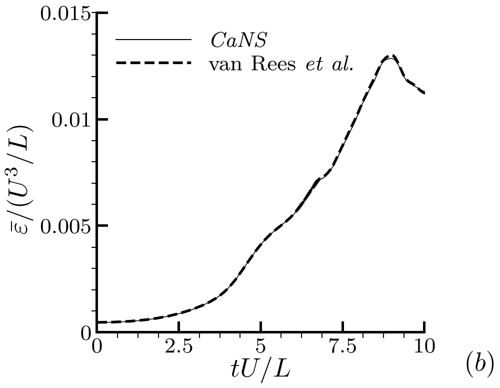


Fig. 6. (a) Visualization of iso-surfaces of vorticity magnitude $|\boldsymbol{\Omega}|$ for the Taylor–Green vortex benchmark. The left panel shows the initial condition with $|\boldsymbol{\Omega}| = 1.5U/L$ and the right panel with $|\boldsymbol{\Omega}| = 15U/L$ at instant $t = 9L/U$. The colors correspond to the z-component of vorticity (see the legend). (b) Temporal evolution of the mean viscous dissipation $\bar{\varepsilon}$ of a three-dimensional Taylor–Green vortex, compared to the reference data from a pseudo-spectral code in [36].

4.1.3. *Taylor–Green vortex*

The last validation considers the temporal evolution of a Taylor–Green vortex. The flow is solved in a tri-periodic domain with dimensions $[0, 2\pi]^3$, with the following initial condition for the velocity field $\mathbf{u}(\mathbf{x}, t = 0) = (u_0, v_0, w_0)$:

$$u_0 = U\sin(x/L)\cos(y/L)\cos(z/L), \tag{6}$$

$$v_0 = -U\cos(x/L)\sin(y/L)\cos(z/L), \tag{7}$$

$$w_0 = 0\,. \tag{8}$$

with $U = 1, L = 1$, and a Reynolds number $\mathrm{Re}_{TG} \equiv UL/\nu = 1600$. Other computational parameters are shown in Table 3.

In this case, a smooth initial velocity field will produce vorticity due to vortex-stretching, generating small-scale vortical structures [35]. This mechanism can be visualized in Fig. 6(a), which shows iso-contours of vorticity magnitude for the initial condition and the instant corresponding to the maximum value of energy dissipation.

The generation of small-scale vortical structures leads to a net increase in viscous dissipation of kinetic energy, $\varepsilon = 2\nu\mathbf{S} : \mathbf{S}$; with $\mathbf{S}$ being the strain-rate tensor $\mathbf{S} \equiv (\nabla\mathbf{u} + \nabla\mathbf{u}^T)/2$. For larger times, this quantity will show a net decrease and eventually vanish, since there is no external power input. Fig. 6(b) shows the expected trend, with the mean (i.e. space-averaged) viscous dissipation reaching a maximum at $t = 9L/U$. The results are compared to the reference data in [36], showing good agreement.

4.2. *Computational performance*

Finally, we test the scaling performance of our implementation for a tri-periodic domain. Since the DFT is cheaper than the other DT, this case corresponds to the highest memory bandwidth per FLOP. It should be, therefore, the worst case scenario for a scaling test.

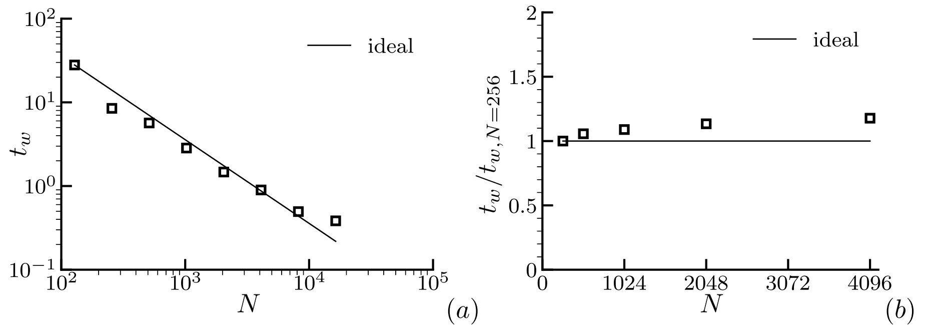


Fig. 7. (a) Strong scaling of the numerical method up to 16 384 cores in a domain with 1024^3 grid cells. t_w denotes wall-clock time in seconds, and N the number of cores. (b) Weak scaling performance for a domain with $1024^3/512 \approx 2 \cdot 10^6$ grid cells per core. t_w denotes mean wall-clock time in seconds/time step/task (i.e. three Runge–Kutta substeps), and N the number of cores. $t_{w,256}$ corresponds to the wall-clock time for $N = 256$.

The simulations were performed in partition A2 (Knights Landing) of the supercomputer MARCONI from Cineca, Italy. Only distributed-memory parallelization was tested in the present work, and sufficed for achieving good scaling. The shared-memory implementation may be useful for future extensions, but as implemented now does not seem to improve the performance. The strong scaling tests were performed in a domain with 1024^3 grid cells. Fig. 7(a) shows the wall-clock time versus the number of cores. Superlinear speedup can be noticed for a smaller number of cores, likely due to cache effects. Overall, the code shows very good scaling performance up to about $O(10^4)$ cores, reaching a small wall-clock time $t_w \approx 0.5$ s/core/timestep. These numbers are consistent with the good performance of the 2DECOMP&FFT library [20], which handles the most demanding parallelization steps.

Weak scaling tests are illustrated in Fig. 7(b), with the number of grid cells per task fixed to $2 \cdot 10^6$. The results show a slight monotonic deterioration of up to 17% from $N = 256$ to 4096 cores. Overall, the results indicate that strong scaling timings are likely to scale up to much larger problem sizes.

5. Conclusions and outlook

We presented an efficient numerical algorithm for massively-parallel DNS of canonical turbulent flows. The method uses a direct, FFT-based solver for the pressure Poisson equation discretized with second-order central differences, parallelized with a 2D *pencil*-like domain decomposition. This approach has been applied recently to massively-parallel numerical simulations of complex turbulent flows with $O(10^9)$ spatial degrees of freedom, but restricted to at least two periodic directions; see e.g. [21–23]. To the best of our knowledge, this is the first general implementation of such parallel algorithms allowing for the different combinations of homogeneous pressure boundary conditions, that can benefit from the method of eigenfunction expansions. Our approach was shown to scale up to about 10^4 cores for a problem with 10^9 spatial degrees of freedom, reaching a very small wall-clock time. These figures will probably scale for larger problem sizes.

The method was validated against distinct benchmark cases of canonical laminar and turbulent flows. It should be noted that several other configurations could have been considered. Obvious examples are wall-bounded flows with inflow/outflow boundary conditions.

For low Reynolds number flows (or extremely high resolution), implicit temporal integration of the diffusion term can be advantageous. In that case, non-staggered discrete transform operators can be considered for non-periodic cases, but the velocity boundary conditions in x and y must be homogeneous. This has been implemented in our numerical tool.

This type of Navier–Stokes solvers, combined with other methodologies to handle, e.g., complex geometries or multiphase flows, have been unveiling important physical insights into flows that require massively-parallel DNS. In the same spirit, the resulting open-source code, more than a tool for simulations of canonical flows, can be seen as an efficient base solver on top of which numerical methods for more complex flows can be implemented.

Acknowledgments

Wim-Paul Breugem and Bendiks Jan Boersma from the TU Delft are acknowledged for interesting discussions. Also, the structure of the open-source DNS code has been inspired by the in-house DNS codes used in their groups. Vilborg Gudjónsdóttir from TU Delft is thanked for proof-reading this manuscript.

Marco Rosti and Mehdi Niazi from KTH Mechanics are thanked for providing the initial conditions for the turbulent duct and channel flow simulations, respectively.

The implementation ran on three different systems. We acknowledge the computing time in the Dutch National Supercomputer Cartesius granted by NWO (Netherlands Organisation for Scientific Research), Netherlands, for first development tests. The validation simulations were performed on the supercomputer Beskow at the PDC center, KTH, with computing time provided by SNIC (Swedish National Infrastructure for Computing), Sweden. Finally, the supercomptuer MARCONI from Cineca in Italy was used under a PRACE, Belgium grant for the scaling simulations.

References

[1] T. Ishihara, T. Gotoh, Y. Kaneda, Study of high–Reynolds number isotropic turbulence by direct numerical simulation, Annu. Rev. Fluid Mech. 41 (2009) 165–180.
[2] S.A. Orszag, G. Patterson Jr., Numerical simulation of three-dimensional homogeneous isotropic turbulence, Phys. Rev. Lett. 28 (2) (1972) 76.
[3] J. Kim, P. Moin, R. Moser, Turbulence statistics in fully developed channel flow at low Reynolds number, J. Fluid Mech. 177 (1987) 133–166.
[4] S. Gavrilakis, Numerical simulation of low-Reynolds-number turbulent flow through a straight square duct, J. Fluid Mech. 244 (1992) 101–129.
[5] A. Vreman, J.G. Kuerten, Comparison of direct numerical simulation databases of turbulent channel flow at Re τ = 180, Phys. Fluids 26 (1) (2014) 015102.
[6] G.L. Kooij, M.A. Botchev, E.M. Frederix, B.J. Geurts, S. Horn, D. Lohse, E.P. van der Poel, O. Shishkina, R.J. Stevens, R. Verzicco, Comparison of computational codes for direct numerical simulations of turbulent Rayleigh–Bénard convection, Comput. & Fluids (2018).
[7] P. Orlandi, Fluid flow phenomena: a numerical toolkit, vol. 55, Springer Science & Business Media, 2012.
[8] E. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, J. Comput. Phys. 161 (1) (2000) 35–60.
[9] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239–261.
[10] S. Leonardi, P. Orlandi, R. Smalley, L. Djenidi, R. Antonia, Direct numerical simulations of turbulent channel flow with transverse square bars on one wall, J. Fluid Mech. 491 (2003) 229–238.
[11] W. Breugem, B. Boersma, R. Uittenbogaard, The influence of wall permeability on turbulent channel flow, J. Fluid Mech. 562 (2006) 35–72.
[12] M. Uhlmann, An immersed boundary method with direct forcing for the simulation of particulate flows, J. Comput. Phys. 209 (2) (2005) 448–476.
[13] R. Verzicco, R. Camussi, Numerical experiments on strongly turbulent thermal convection in a slender cylindrical cell, J. Fluid Mech. 477 (2003) 19–49.
[14] G.H. Golub, C.F. Van Loan, Matrix computations, vol. 3, JHU Press, 2012.
[15] R.B. Wilhelmson, J.H. Ericksen, Direct solutions for Poisson's equation in three dimensions, J. Comput. Phys. 25 (4) (1977) 319–331.
[16] P.N. Swarztrauber, The methods of cyclic reduction, Fourier analysis and the facr algorithm for the discrete solution of Poissons equation on a rectangle, SIAM Rev. 19 (3) (1977) 490–501.
[17] U. Schumann, R.A. Sweet, Fast Fourier transforms for direct solution of Poisson's equation with staggered boundary conditions, J. Comput. Phys. 75 (1) (1988) 123–137.
[18] J. Adams, P. Swarztrauber, R. Sweet, FISHPAK: A package of Fortran subprograms for the solution of separable elliptic partial differential equations,The National Center for Atmospheric Research, Boulder, CO 1980.
[19] M.S. Dodd, A. Ferrante, A fast pressure-correction method for incompressible two-fluid flows, J. Comput. Phys. 273 (2014) 416–434.
[20] N. Li, S. Laizet, 2decomp & fft-a highly scalable 2d decomposition library and fft interface, in: Cray User Group 2010 Conference, 2010, pp. 1–13.
[21] R. Ostilla-Mónico, R. Verzicco, S. Grossmann, D. Lohse, The near-wall region of highly turbulent Taylor–Couette flow, J. Fluid Mech. 788 (2016) 95–117.
[22] M.S. Dodd, A. Ferrante, On the interaction of Taylor length scale size droplets and isotropic turbulence, J. Fluid Mech. 806 (2016) 356–412.
[23] P. Costa, F. Picano, L. Brandt, W.P. Breugem, Universal scaling laws for dense particle suspensions in turbulent wall-bounded flows, Phys. Rev. Lett. 117 (13) (2016) 134501.
[24] E.P. van der Poel, R. Ostilla-Mónico, J. Donners, R. Verzicco, A pencil distributed finite difference code for strongly turbulent wall-bounded flows, Comput. & Fluids 116 (2015) 10–16.
[25] V. Fuka, Poisfft–a free parallel fast Poisson solver, Appl. Math. Comput. 267 (2015) 356–364.
[26] A.A. Amsden, F.H. Harlow, A simplified MAC technique for incompressible fluid flow calculations, J. Comput. Phys. 6 (1970) 322–325.
[27] P. Wesseling, Principles of computational fluid dynamics, vol. 29, Springer Science & Business Media, 2009.
[28] R. Verzicco, P. Orlandi, A finite-difference scheme for three-dimensional incompressible flows in cylindrical coordinates, J. Comput. Phys. 123 (2) (1996) 402–414.
[29] M. Frigo, S.G. Johnson, The design and implementation of fftw3, Proc. IEEE 93 (2) (2005) 216–231.
[30] E. Anderson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammarling, J. Demmel, C. Bischof, D. Sorensen, LAPACK: A portable linear algebra library for high-performance computers, in: Proceedings of the 1990 ACM/IEEE Conference on Supercomputing, IEEE Computer Society Press, 1990, pp. 2–11.
[31] A. Samarskij, E. Nikolajev, Numerical methods for grid equations. Vol. I: Direct methods, Birkhuser, Basel Boston Berlin, 1989.
[32] H.C. Ku, R.S. Hirsh, T.D. Taylor, A pseudospectral method for solution of the three-dimensional incompressible Navier-Stokes equations, J. Comput. Phys. 70 (2) (1987) 439–462.
[33] D.S. Henningson, J. Kim, On turbulent spots in plane Poiseuille flow, J. Fluid Mech. 228 (1991) 183–205.
[34] S.B. Pope, Turbulent flows, Cambridge University Press, 2000.
[35] M.E. Brachet, D.I. Meiron, S.A. Orszag, B. Nickel, R.H. Morf, U. Frisch, Small-scale structure of the Taylor–Green vortex, J. Fluid Mech. 130 (1983) 411–452.
[36] W.M. Van Rees, A. Leonard, D. Pullin, P. Koumoutsakos, A comparison of vortex and pseudo-spectral methods for the simulation of periodic vortical flows at high Reynolds numbers, J. Comput. Phys. 230 (8) (2011) 2794–2805.