ARTICLE IN PRESS

Computers and Mathematics with Applications ▮ (▮▮▮▮) ▮▮▮–▮▮▮

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

ARTICLE INFO

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

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

ABSTRACT

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/

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,

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

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

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 $n + 1$):

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

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

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

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

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

Please cite this article in press as: P. Costa, A FFT-based finite-difference solver for massively-parallel direct numerical simulations of turbulent flows, Computers and Mathematics with Applications (2018), https://doi.org/10.1016/j.camwa.2018.07.034.