A CLASSICAL SPECTRAL, COMPACT-DIFFERENCE DIRECT NUMERICAL SIMULATION OF TURBULENT FLOW IN 300 LINES OF CPL

Paolo Luchini and Maurizio Quadrio
CPLcode.net

September 20, 2022

Abstract

We describe the code `scddns` (Spectral, Compact-Difference DNS), a short yet self-contained implementation of our numerical method for the Direct Numerical Simulation (DNS) of a turbulent channel flow, that is in fact suitable for any wall-bounded flow with two homogeneous directions. The method follows the traditional pseudo-spectral approach, with Fourier discretization in the wall-parallel (periodic) homogeneous directions. Its distinctive feature is a colocated discretization method (compact, 4th-order finite differences) for the wall-normal direction, thus enabling minimal memory footprint and reduced communication for parallel computing.

1 Introduction

The CPL implementation was conceived in 1999, first as a parallel version for shared-memory architectures, and shortly thereafter as a distributed-memory version that avoids the conventional message-passing paradigm implemented via the MPI library. The code, documented first in [2], was then recently extended to also work under both conventional OpenMP and MPI environments. A companion version in cylindrical coordinates is also available. The full version `scddns` is parallel, contains equations for a passive scalar, a freely tunable body force, and a streamlined framework to assign generic (steady or time-dependent) boundary conditions and forcing terms. Here three versions are included: directory `simple` contains the minimalist serial version that can more easily be used to

study the algorithm; directories `OpenMP` and `MPI` contain the respective parallel versions.

2 Description of the algorithm

The incompressible Navier–Stokes equations are formulated so that continuity and momentum equations in primitive variables are replaced by two scalar equations, one (second-order) for the normal component of vorticity and one (fourth-order) for the normal component of velocity, much in the same way as the Orr–Sommerfeld and Squire decomposition of linear stability problems. The main advantages of this approach are that i) pressure is eliminated from the equations, and ii) the two wall-parallel velocity components are recovered as the solution of a 2×2 algebraic system (a computationally cheap procedure), when a Fourier expansion is adopted for the homogeneous directions.

Discretization is indeed Fourier in the wall-parallel directions, but spectral methods are avoided in the discretization of wall-normal y derivatives. To minimize parallel communication, which is essential in situations where the bandwidth among computing elements is limited, compact finite-difference schemes are used. This can be advantageous if one wants to avoid high-performance network fabric and use conventional CPUs on distributed-memory machines. Nowadays, the same approach is gaining interest to link together different GPUs, the communication among which is relatively slower. A five-point stencil is used, which yields (internally computed) compact formulas for the first-, second- and fourth-derivative operators appearing in the equations of motion. The coefficients d_1^j of, say, the first derivative are then used to compute an approximated first derivative of a function $f(z)$ at a node z_j as:

$$f'(z_j) \simeq \sum_{i=-2}^{2} d_1^j(i) f(z_{j+i})$$

Temporal discretization is quite flexible, within the chosen design of a partially-implicit method where the stability-limiting viscous terms are integrated implicitly, whereas the convective terms are integrated explicitly and can benefit from a higher-accuracy scheme. The implicit scheme is always the second-order Crank-Nicolson, while the explicit scheme, simply defined via its coefficients, can be changed at will by first defining the scheme with a two-lines subroutine, and then by using that subroutine in the time-stepping procedure. The `channel-simple` version only contains a workhorse scheme, made by a three-substeps low-storage Runke–Kutta scheme, but the complete `channel` version includes a small library of possible schemes: writing the name of the chosen routine within the main time integration loop is all it takes to switch to alternate integration schemes.

The minimal memory footprint is another advantage of choosing finite differences for wall-normal discretization. The classic three-substeps Runke–Kutta needs to store two complete flow fields (one at current time and one at the previous time), so that the code needs only five variables per grid point: the current velocity field, and a two complete scalar fields (wall-normal velocity and vorticity) at the past time level.

3 CPL implementation

The version `simple/scddns` runs on a single core, integrates the incompressible Navier–Stokes equations set up for the canonical indefinite plane channel flow without passive scalar and body forces. The streamwise, spanwise and wall-normal directions are x, y and z.

The main building blocks are the following:

- the initial setup of the FD coefficients for the wall-normal derivatives within the `MODULE setup_derivatives`; they are computed once and for all, by solving a linear system which enables a straightforward management of unequally spaced mesh, and stored in the one-dimensional structure `derivatives`; the boundary values are adjusted next, and eventually compact functions like `D1()` are defined for later streamlined use;

- definition of data structures and auxiliary functions; one notable example is the velocity array, defined as `VELOCITY=STRUCTURE(COMPLEX u,v,w); ARRAY(0..nx,-ny..ny,-1..nz+1) OF VELOCITY V`. The Fourier modes are `2*nx+1` and `2*ny+1`, only half of them are actually stored thanks to hermitian symmetry. The variable `maxtimelevels` controls the memory allocation required by the time integration scheme, chosen via the `SUBROUTINE timescheme`;

- `SUBROUTINE convolutions` computes the convolution of Fourier modes with the pseudo-spectral approach, by removing aliasing error with the 2/3 rule;

- `SUBROUTINE builrhs` builds the right-hand-side of the linear system and takes advantage of the (inlined for efficiency) `FUNCTION OS()` and `SQ()` to compactly write the equations of motion, which are the fully nonlinear counterpart of the Orr-Sommerfeld and Squire stability equations;

- `SUBROUTINE linsolve` solves the linear system, and lays the foundation of its parallel solution via the modified Gauss algorithm;

- the chosen temporal integration scheme is used in its generic form in `buildrhs`; selecting a specific scheme is easily done by the proper call(s) in the main temporal integration loop, e.g. with `buildrhs(RK1_rai)`.

4 Usage

A simulation is fully defined by the parameter values assigned via the `scddns0.dat` input file. Here the user selects discretization parameters via the integers `nx,ny,nz`: `nx` and `ny` are half the number of modes in each homogeneous direction, and arrays in the wall-normal direction extend from the ghost node at `iz=-1` to the ghost node at `iz=nz+1`, with walls at `iz=0` and `iz=nz`. Moreover, the domain size $L_x = 2\pi/\alpha_0$ and $L_z = 2\pi/\beta_0$ is set by the parameters `alpha0` and `beta0`; the reference length is always assumed to be one half of the gap, which is thus bound to be 2 in dimensionless form. The parameter `htcoeff` varies the one-dimensional mesh compression near the solid walls. The value of the Reynolds number Re is also prescribed here. While the reference length necessary to provide a meaning to the numerical value of Re is hard-coded in the prescribed $L_z = 2$, the choice of the forcing term decides on the reference velocity scale. The simulation can be driven at Constant Flow Rate [3], by selecting either `meanflowx=2` (bulk Reynolds number) or `meanflowx=1.3333333` (Poiseuille Reynolds number, where the centerline velocity is that of a laminar Poiseuille flow with the same flow rate), or at a Constant Pressure Gradient, by selecting `meanpx=1` (friction Reynolds number). The time step size `delta_t` and total duration of the simulation `t_max` complete the list of discretization parameters, and the parameter `dt_save` sets the time interval at which a new snapshot is saved to the database for further statistical analysis. Snapshots have the same form as the input file, and any of them can be used to restart the computation. An optional initial velocity field is provided immediately after the simulation parameters, or can be copied from another input file of the same format denoted by the parameter `Vfield`.

The relatively trivial laminar Poiseuille flow can be set up with e.g. `nx=ny=4` and e.g. `nz=100`, by selecting `meanflowx=2`, any value of Re and empty velocity field.

Simulating a turbulent case requires either an initial flow field, or to start from a perturbed laminar parabolic profile; perturbations are added with amplitude `eps` to be defined in the source, when no other input field is given.

The output printed on screen informs of the simulation time (column 1), the wall-shear stress at the two walls (columns 2,3), the driving pressure gradient (column 4) and the resulting flow rate (column 5).

5 Example

The provided version of the parameter file `scddns0-Re180Px-32.dat` enables one to reproduce the case described by Kim, Moin and Moser in 1987 [1], the famous first DNS of a plane channel flow. To do it, just compile the code and run it.

The reproduction is meant to be qualitative only. The discretization of the provided initial field is not the same (to yield a short computing time with a single core), and in particular the size of the computational domain is reduced, while the spatial resolution is still a bit low. If needed, the initial field – which includes 64^2 modes and 100 points in the wall-normal direction – can be seamlessly reinterpolated into a larger/smaller grid, by simply changing the values of `nx` and `ny` in the parameter file. In this version, the number of points in the wall-normal direction cannot be changed. Computing time can be adjusted at will by tweaking the variable `t_max`.To compute statistical quantities requires to build first a database (velocity fields written at a preselected period defined by the parameter `dt_field`) to be post-processed later. A small utility `postprocess.cpl` is provided to post-process the database and to extract mean velocity profiles and Reynolds stresses.

As in the original paper, the simulation employs a Constant Pressure Gradient, and the reference Reynolds number is $Re_\tau = 180$, based on the friction velocity u_τ and the distance h, half the gap between the two walls. The steps to obtain statistics comparable to those of Ref.[1] are as follows:

- enter directory simple;
- compile `scddns.cpl`, and run it without changing the parameters;
- the set values of `delta_t=0.002` and `t_max=10` provide for a small enough timestep to grant stability, and an integration time equal to that of Ref.[1] to obtain reasonable statistics;
- the set value `dt_field=0.5` dictates the time interval at which a full field is stored for further analysis, and provides a reasonable compromise between number of samples and storage requirements;
- after the simulations is completed, compile `postprocess.cpl` and run it for other statistics;
- a single time step (made by three Runge–Kutta steps) takes approximately 0.45 seconds on a single core of an old CULV Intel Core M-5Y71 CPU, a

rather outdated hardware. This correponds to approximately 3.6×10^{-7} seconds per point and time(sub)step. This figure might legitimately be translated into 1.5×10^{-7} seconds, by considering the expansion on the larger number `nxd, nyd` of modes to eliminate aliasing errors.

References

[1] J. Kim, P. Moin, and R. Moser. Turbulence statistics in fully developed channel flow at low Reynolds number. *J. Fluid Mech.*, 177:133–166, 1987.

[2] P. Luchini and M. Quadrio. A low-cost parallel implementation of direct numerical simulation of wall turbulence. *J. Comp. Phys.*, 211(2):551–571, 2006.

[3] M. Quadrio, B. Frohnapfel, and Y. Hasegawa. Does the choice of the forcing term affect flow statistics in DNS of turbulent channel flow? *Eur J Mech B Fluids*, 55:286–293, 2016.