Computers & Fluids 39 (2010) 471–484

Contents lists available at ScienceDirect

Computers & Fluids

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

A numerical strategy to combine high-order schemes, complex geometry and parallel computing for high resolution DNS of fractal generated turbulence

S. Laizet [a,b,*], E. Lamballais [b], J.C. Vassilicos [a]

[a] Turbulence, Mixing and Flow Control Group, Department of Aeronautics and Institute for Mathematical Sciences, Imperial College London, London SW7 2AZ, United Kingdom
[b] Laboratoire d'Etudes Aérodynamiques UMR 6609, Université de Poitiers, ENSMA, CNRS, Téléport 2 - Bd. Marie et Pierre Curie B.P. 30179, 86962 Futuroscope Chasseneuil Cedex, France

ARTICLE INFO

Article history:
Received 25 September 2008
Received in revised form 12 August 2009
Accepted 30 September 2009
Available online 13 October 2009

ABSTRACT

Impressive advances in parallel platform architectures over the past decade have made Direct Numerical Simulation (DNS) a powerful tool which can provide full access to the spatial structure of turbulent flows with complex geometries. An innovative approach which combines high-order schemes and a dual domain decomposition method is presented in this paper and is applied to DNS of multiscale-generated turbulent flows by a fractal grid. These DNS illustrate the applicability of our approach to the simulation of complex turbulent flows and provide results which are compared with recent laboratory experiments thus providing new insights for the interpretation of the experimental measurements.

1. Introduction

Arguably, fluid turbulence is the most important and most difficult open problem in mechanics and presents an unprecedented challenge for the scientific community. The combination of unsteadiness, non-linearity (self-advection) and non-locality (pressure fluctuations) at comparable measures makes it a formidable problem which has resisted impressive attempts from some of the best 20th century theorists. It seems that all available theoretical and mathematical tools have been tried and have failed. However, progress in the understanding of turbulence has been made over the past 40 years, and much if not most of it is accountable to the advent of computers and advances in numerical methods (e.g. the Fast Fourier transform).

The turbulence problem will be solved when some combination of theoretical/mathematical and numerical approaches will be found which can be used to work out the turbulent flow properties in any imaginable configuration (aeronautical, environmental, geophysical, chemical, nuclear, etc.) without the need for laboratory experiments, whether scaled down or not. It will be a very important quantum leap of progress even if such a solution method is to be found for some, or even only one, type of applications, such as all possible/imaginable aerodynamic shapes, or all possible/imaginable combustion chambers, etc. This statement of what constitutes a solution to the turbulence problem is eminently practical and only makes sense if the cost and the time to be used on computers is significantly less than the cost of and the time for carrying out the experiments. This sets a trivial limit on the computational resources to be expended in such solutions.

Some may think that there is no need to find a solution to the turbulence problem because industry has already empirically solved most of its technological problems and has perhaps evolved, over the years, technological solutions which are close to the best possible. However, times are changing and will always be changing; now, for example, climate and environmental urgencies as well as the need to find new, cheaper, energy sources require new cleaner and less energy-consuming solutions. These new technological solutions (new industrial mixers, new transport vehicles, new combustors, etc.) will require new flow concepts and it will be too expensive (perhaps unaffordable) to work them out without a solution to the turbulence problem of the kind described above.

It is therefore, important to start developing numerical strategies for the solution of the Navier–Stokes equations without modelling, with high accuracy and with industrial-like versatility, i.e. with the possibility to easily modify boundary and initial conditions and allow for as much complexity as possible. Currently, as well as in the foreseeable future, numerical simulations will require extraordinarily powerful resources.

As it calculates explicitly the spatio-temporal variation of most of the significant energy-carrying scales, Direct Numerical Simulation (DNS) is recognized to be the most accurate computational approach to fluid turbulence. Unfortunately, the computational cost of DNS is very high due to the very high number of degrees of freedom involved in realistic turbulence, as a result of the non-linearity of the Navier–Stokes equations. For flows with relatively complex geometries at relevant Reynolds number (i.e. representative of real situations), the computational resources required by DNS often drastically exceed the capacity of the most powerful

* Corresponding author. Address: Turbulence, Mixing and Flow Control Group, Department of Aeronautics and Institute for Mathematical Sciences, Imperial College London, 53 Prince's gate, London SW7 2AZ, United Kingdom.
E-mail addresses: s.laizet@imperial.ac.uk (S. Laizet), eric.lamballais@univ-poitiers.fr (E. Lamballais), j.c.vassilicos@imperial.ac.uk (J.C. Vassilicos).

0045-7930/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compfluid.2009.09.018

supercomputers. However, the massive parallelisation strategy proposed by the new generation of supercomputers can now make previously unreachable DNS of realistic turbulent flow configurations just about affordable.

Very large DNS of turbulent flow have indeed started to occur after the turn of this century. Mitsuo Yokokawa et al. [22] have been able to perform on the Earth Simulator in Japan a DNS of 3D homogeneous turbulence at high Reynolds number in a tri-periodic computational domain using 2048^3 Fourier modes. More recently, Hoyas and Jimenez [6] have computed a high Reynolds number turbulent channel flow (in a bi-periodic computational box) using 2048 computational cores and involving more than 35 Tb of data. The computational challenge is both in the actual simulations as well as in the post-processing analysis, the latter being particularly stringent when a detailed physical analysis is conducted on an extremely large amount of data.

Following these very large DNS, the second step which needs to be taken is the possibility of implementing complicated and/or non-academic flow configurations. This can now be attempted by an efficient use of parallel-architecture platforms, following the continual increase in number of computational cores for each new generation of parallel-architecture platforms. This stage requires to choose a relevant numerical code and to carry out its adaptation without spoiling its initial qualities of accuracy, efficiency or simplicity.

In this paper, we are interested in this new opportunity of using massive parallel-architecture platforms to compute, by DNS, fluid turbulence generated by complex geometries. Our purpose is to define a favourable combination of numerical methods and computational techniques and apply them to flows that were numerically unreachable until very recently. Concerning the numerical methods, the present choice is in high-order schemes with an accuracy preserved through the use of a regular computational mesh. The corresponding code (called **Incompact3d**) selected in this work is an intermediate tool between fully spectral Navier–Stokes solvers for academic geometry (Fourier/Chebyshev representation) and more versatile codes based on standard numerical schemes of lower accuracy. To allow the treatment of complex geometries despite the Cartesian mesh, it is proposed to use an Immersed Boundary Method (IBM). The aim of the present work is to evaluate how this type of numerical code can easily and efficiently be adapted to a massive parallel-architecture platform in order to address turbulent flow problems where the number of degrees of freedom is more than one order of magnitude larger than is currently affordable with the most powerful vector-architecture platforms.

To demonstrate the potential of our approach, we choose to apply it to turbulence generated by multiscale (fractal) grids because of the complexity of the grid and also because results from such laboratory experiments have been recently published [7,19] and can be used to make some comparisons with our computations. We therefore, present in this paper high resolution DNS of turbulent flows generated by the fractal grids of Fig. 1.

Physically, the fundamental understanding of multiscale flow dynamics resulting from turbulence generation by multiscale objects is necessary, with interesting possibilities for use in aerodynamic noise applications, mixing applications in the chemical process and other industries, etc. For this, a well-designed and well-targeted canonical experiment has been proposed recently at Imperial College London where the first experiments of turbulence generated by fractal grids in wind tunnels have been conducted with interesting and unexpected results [7,19]. To highlight the more surprising findings of these experimental studies, it is necessary to have full access to the spatial structure of the mean and instantaneous flows. In this paper, with the help of the new generation of parallel-architecture platforms, it is suggested that, despite the very large runs required, DNS might be able to provide information in a way that is complementary to advanced experimental techniques.

The paper is organised as follows. After a brief description of the numerical methods in Section 2, the technique used to parallelise the code is described in Section 3. The performances obtained with this new code are presented in Section 4. Results from simulations of fractal generated turbulence and comparisons with experiments are detailed in Section 5. Conclusions and future directions are summarized in Section 6.

2. General numerical method

For the present flow configuration, three requirements need to be combined by a relevant choice of the numerical method: (i) high accuracy, (ii) ability for complex geometries and (iii) efficiency and portability to massively parallel architectures.

By the first requirement, it is important to accurately compute a fully turbulent flow without introducing any bias in the small scale dynamics, such as artificial anisotropy errors caused by the computational mesh. To avoid this type of non-physical effects, accurate schemes with very low dissipative and dispersive errors must be used. It is known that the most efficient choice for this is a fully spectral approach. However, such an approach constrains the choice of flow configuration, and the typical computational configuration used in academic studies of turbulence is the tri-periodic box (treated by Fourier expansion) based on a Cartesian collocated mesh. For slightly more realistic geometries, such a fully spectral approach in Fourier or Chebyshev space becomes unfortunately inoperable. To allow the use of a computational mesh matching a complex geometry while preserving the spectral accuracy, recent developments on the spectral element method (see the book of [3] for a review in the context of incompressible flows) seem to be promising. However, the use of high-order schemes with a sophisticated computational mesh remains in general very demanding in terms of code development, the price to pay to reach

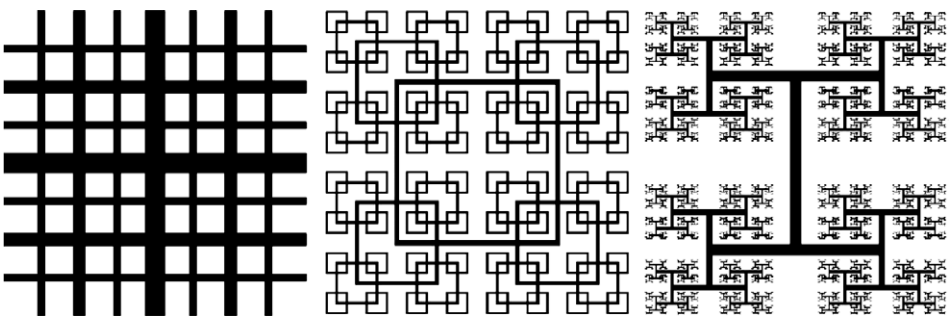

Fig. 1. Scaled diagrams of a fractal cross grid (left), a fractal square grid (middle) and a fractal I grid (right).

high accuracy without a collapse of the computational efficiency. As an alternative to this ambitious strategy, we suggest to take advantage of the low computational cost of finite difference schemes to achieve high resolution DNS through a different compromise much easier to implement while providing a high computational efficiency. In our flow configurations where inflow/outflow boundary conditions are present in one direction, we propose to use a code based on sixth-order compact finite-difference schemes [14], thereby allowing straightforward treatment of these boundary conditions while mimicking the behaviour of spectral methods via the so-called spectral-like accuracy. Note in particular that compact schemes drastically limit the anisotropy errors [14] associated with the computational mesh organisation, because of their very low dispersion error over a wide range of scales. In brief, the use of compact finite-difference schemes on a Cartesian mesh can be seen as a numerical method that is close to the spectral one with only moderate loss of accuracy which is, however, compensated by the possibility to treat more general boundary conditions than just periodicity or free-slip.

The second requirement is the ability of the numerical method to accurately describe the very complex geometry of the fractal grid. In conventional Computational Fluid Dynamics (CFD), especially in an industrial context, complex geometries are treated using non-structured element meshes, requiring low-order schemes and sophisticated tools for the generation of highly distorted meshes. The resulting accuracy would be clearly incompatible with our primary goal of a sharp description of all the relevant scales of the generated turbulence. Here, to preserve the use of highly accurate schemes on a simple Cartesian mesh, we use an Immersed Boundary Method where the body's presence is mimicked by the introduction of a local forcing in the governing equation. More precisely, we use in this study a direct approach where the fluid's motion is halted inside the fractal body by an instantaneous adjustment of the forcing which thereby ensures the no-slip condition at the surface. For more details about this technique combined with the present numerical schemes, see [16,15]. Note that we use here a simplified version of the direct forcing approach for which the target velocity in the immersed region is simply zero. A more sophisticated technique can be used where the forcing term is defined to create an artificial flow inside the body in order to improve the regularity of the velocity field across the modelled surface [16,15]. However, for the present complex geometry which is located upstream of the vortical region of the flow, the numerical development required to impose such an artificial flow has not been judged useful on account of the small benefits which would be expected in terms of accuracy for the description of the vortex dynamics further downstream. In fact, to the knowledge of the authors, a better accuracy than second-order has never been shown in the literature in the framework of IBM. The present code's behaviour in presence of IBM is consistent with this limitation, with at the best second-order accuracy observed in academic benchmarks [10]. This observation might suggest that the use of high-order schemes is useless, second-order schemes being a priori sufficient. This assertion is right formally (in terms of asymptotic convergence) but misleading practically, as shown for instance in [16,15]. In agreement with these previous works, the accuracy in the neighbourhood of the fractal grid is not optimal, but further downstream, a clear benefit of the use of compact schemes can be expected through the accurate description of the developing turbulence.

The third requirement, which is related to the computational efficiency, is the object of the developments and validations described in the following sections. The code which is adapted here to MPP (Massive Parallel Processing) is called **Incompact3d**. Initially, this code was developed for vector-architecture platforms. It solves the incompressible Navier–Stokes equations using sixth-order compact schemes for the spatial discretization. For the time integration, different schemes can be used (Adams–Bashforth or Runge–Kutta schemes) depending on the flow configuration. Even if Runge–Kutta schemes (third- or fourth-order) should improve the temporal accuracy and the numerical stability, a second-order Adams–Bashfort scheme is used in the present study. Indeed, our experience is that when combined with an Immersed Boundary Method (IBM), the use of sub-time steps (inherent to Runge–Kutta schemes) can lead to a slight deterioration of the accuracy of the no-slip condition (at the boundary of the solid body) ensured by the direct forcing. To discretize the parallelepipedal computational domain $L_x \times L_y \times L_z$, a Cartesian mesh of $n_x \times n_y \times n_z$ mesh points is used, with a regular distribution in the three spatial directions. Note, however, that **Incompact3d** offers the possibility of a local refinement in one direction using a mapping technique proposed by [2,1]. Inflow/outflow boundary conditions are used in the streamwise direction x while periodicity is assumed in the two other directions (many other combinations of boundary conditions are available in the code). To ensure the incompressibility condition, a fractional step method is used, requiring the solution of a Poisson equation for the pressure. Following the technique of [21,20,8] extended to high-order schemes on a stretched mesh, the Poisson equation is fully solved in spectral space using Fast Fourier transform routines. It should be emphasized that a Fourier representation is used in the periodic $(y-z)$ directions as well as in the inflow/outflow direction (x) through the relevant use of cosine expansions. Combined with the concept of the modified wave number, this direct (i.e. non-iterative) technique allows the implementation of the divergence-free condition up to machine accuracy while also allowing the use of a stretched mesh in one direction (even if this option is not exploited for the present study). A partially staggered mesh is used where the pressure mesh is shifted by a half-mesh from the velocity mesh in each direction. This type of mesh organisation leads to more physically realistic pressure fields with no spurious oscillations. For more information regarding this new version of **Incompact3d**, see [10] for technical details and [12,13] for recent results and validations. To conclude this section, note that **Incompact3d** satisfies the first requirement (high accuracy) whilst also being computationally efficient for use on vector-architecture platforms due to the direct nature of its pressure solver. The purpose of this paper is to examine whether this good combination of numerical method and body modelling can be favourably adapted to massively parallel-architecture platforms in order to allow very high resolution DNS of complex turbulent flows.

3. Parallel strategy

Incompact3d is a FORTRAN 90 code that was initially designed for serial processors and later converted to vector processors. The highly vector code is widely used and is limited to simulations with less than 200 million mesh nodes with current vector-architecture platforms.[1] This code remains a very attractive tool for DNS and has recently allowed us to study various original flow configurations [9,12,13]. However, for flows with relatively complex geometry such as those generated by a fractal grid, the computational resources required to simulate the multiscale nature of the flow is found to exceed the capacity of the most powerful vector-architecture platforms currently available. Indeed, the resolution requirement is obviously influenced by the range of scales that need to be accurately represented but also by the numerical method.

[1] Recent powerful vector-architecture platforms (one X2 Cray system and two NEC SX-8 systems) are available at the UK and French National Computer service (HECToR and IDRIS/CCRT, respectively).

Thereby, present multiscale generated turbulent flows cannot be simulated without the aid of state-of-the-art top–end parallel computing which is MPP. More precisely, MPP is a type of computing that uses many separate processors running in parallel to execute a single program. MPP is similar to symmetric processing (SMP for Symmetric Multi Processor), with the main difference that in SMP platforms all the processors share the same memory, whereas in MPP platforms, each processor has its own memory. MPP platforms are, therefore, more difficult to program because the application must be divided in such a way that all the executing segments can communicate with each other. Furthermore, MPP platforms are dedicated to applications using a large number of processors and are the perfect architecture for running our multiscale generated flow simulations.

The parallel version of **Incompact3d** is developed with several objectives: portability of the code (ability for the code to be run on a wide range of platforms), scalability (preservation of the code efficiency when hundreds of processors are used) and conservation of the original structure of the code (direct solver for the Poisson equation and sixth-order compact schemes in the three spatial directions). To ensure portability and scalability, the language independent communications protocol MPI (Message Passing Interface [5]) is used. It is the dominant model in high-performance computing nowadays. Note, however, that OpenMP (Open Multi-Processing), which is a shared-memory parallel programming protocol could also be implemented in conjunction with MPI to provide a second level of parallelism for improved performance on MPP platforms having SMP processors (dual or quadri cores). Programs that mix OpenMP and MPI are often referred to as hybrid codes. To ensure the scalability and keep the same structure of the code, the proposed parallelisation strategy is based on a Domain Decomposition (DD) method: the computational domain is subdivided along one spatial direction into a set of subdomains that can exchange information so as to update each other during the solution process. This strategy is the most general and versatile approach. In this paper, we show that it can lead to excellent parallel efficiency for the present code based on a highly structured (Cartesian) mesh. However, the efficiency of this strategy is also linked to the numerical methods used, especially to their more or less implicit nature.

In **Incompact3d**, the spatial differentiation (derivative and interpolation) is ensured by compact sixth-order finite-difference schemes while an Adams–Bashforth scheme is used for the time advancement. In practice, the explicit nature of the time discretization does not lead to particular problems for the adaptation to parallel computing. Naturally, the stability properties of a given temporal scheme still need to be checked (CFL restriction condition) but without any link to the parallel nature of the code. On the contrary, the spatial discrete operators are implicit in the sense that the evaluation of the derivative/interpolation at one node requires to compute all its counterparts along the direction of differentiation. In the context of parallel computing, this property is found to be very penalizing due to the amount of communication exchange that it requires. The present sixth-order schemes require the inversion of a tridiagonal distributed matrix, this generic problem being highly time consuming because of repetitive communications through the forward and backward dependencies of the computed values node by node. To avoid this problem, one possibility would be to substitute the implicit sixth-order scheme with its explicit same order counterpart, this change being useful only in the direction where the subdomains are divided. However, to guarantee sixth-order accuracy, this explicit scheme requires a stencil of seven mesh nodes [14], which means that all the subdomains must be overlapped at the inner boundaries on a slice of three mesh nodes thickness. These slices, generally known as overlap areas, represent mesh points that contain copies of the boundary values stored on neighbouring subdomains in the mesh topology. During the calculation, the values stored on these slices must be updated receiving values from neighbouring subdomains. As expected, this strategy does not seem well adapted here because large overlap areas are too expensive in terms of communication time when using hundreds of processors. A compromise could be found by using explicit second- or fourth-order explicit schemes in the direction where the subdomains are divided. Despite the fact that this will reduce the overlapping area between each subdomain, the loss of accuracy in one direction could affect the accuracy of the global solution, thus erasing one of the main strengths of **Incompact3d** which is based on high-order schemes with reduced anisotropy errors.

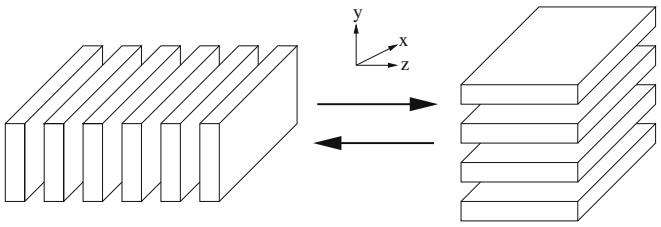

Fig. 2. Schematic illustration of the dual domain decomposition operation.

The parallelisation strategy finally retained is based on a dual domain decomposition. The computational domain is divided into equally sized subdomains in the z-direction, ensuring an equal load balance (Fig. 2, left decomposition). Each MPI-process is then assigned to one subdomain for a simultaneous advancement to the next time level. The algorithm is structured such that no explicit synchronisation statements need to be used (same amount of work for each MPI-process), which means that the values of all variables are available simultaneously for all subdomains. At this stage, no derivative/interpolation in the z-direction can be computed. Then, a global transpose operation is accomplished to get relevant data onto the right MPI-process via the use of a second domain decomposition in the y-direction (Fig. 2, right decomposition). After this operation, calculations can be performed in the z-direction for each new slice. The global computation can then be continued after an inverse transpose operation. These global operations require the parcelling of small blocks of data on each MPI-process, labelling these with the address of the MPI-process that requires the information, transferring the data, reading it by receiving MPI-process, and reconstructing the flow field. Historically, this technique has been used by [18] as a technique derived from practice on the Cray XMP.[2] Indeed, as there was not enough core memory, the data had to be continually swapped into core from fast secondary memory while working on one slice at a time (the Cray XMP had a relatively fast disk that made this feasible).

Note that this parallelisation strategy maintains the original structure of the code since no changes are made in the computation of the spatial differentiations (derivative/interpolation) and in the Poisson solver. Furthermore, as MPI tools and FFTW (Fastest Fourier transform in the West, [24]) are used, the portability of the code is maintained and we have been able to test the code on a wide range of parallel-architecture platforms. In the next section, we present the performance of this new parallel code on four different platforms.

4. Performance evaluation and scalability

To test and validate the new parallel code, several simulations have been conducted on different parallel-architecture platforms:

[2] It was the 1982 successor to the 1976 Cray-1, and the world's fastest computer from 1983 to 1985 with one, two or four processors and up to 128 Mb of memory.

HECToR and HPCx in the United Kingdom, Babel in France and MareNostrum in Spain. These four platforms are, respectively, sitting at positions 29, 261, 9 and 26 in the list[3] of the most powerful platforms in the world in June 2008. The development and the validation of the parallel code have been achieved at Imperial College London.[4] Note also that the migration to each platform has been achieved with very limited effort due to the excellent portability of the code which is preserved despite the present parallel transformation achieved with platform-independent tools and protocols. In this study, the references for speedup and efficiency will be those of one computational core (which corresponds to one MPI-process). Indeed, as most of the processors are based on a dual or quadri core architecture, a speedup based on the number of processors would not make sense.

4.1. Test environment

The HECToR configuration is an integrated system known as "Rainier", which includes a scalar MPP XT4 system and storage systems. The XT4 comprises 1416 compute blades, each of which has 4 dual core processor sockets. This amounts to a total of 11,328 computational cores, each of which acts as a single CPU. The processor is an AMD 2.8 GHz Opteron. Each dual-core socket shares 6 Gb of memory, giving a total of 33.2 Tb in all. The theoretical peak performance of the system is 54.65 T flops.[5] Each dual-core socket controls a Cray SeaStar2 chip router. This has 6 links which are used to implement a 3D-torus of processors. The point-to-point bandwidth is 2.17 Gb/s, and the minimum bi-section bandwidth is 4.1 Tb/s (latency around 6 μs). It is located at the University of Edinburgh, in Scotland.

The HPCx system uses IBM eServer 575 nodes for the computation and IBM eServer 575 nodes for login and disk I/O. Each eServer node contains 16 processors. The main HPCx service provides 160 nodes for compute jobs for users, giving a total of 2560 computational cores. The peak computational power of the HPCx system is 15.3 T flops. The complete new platform gave a value of 12.94 T flops for the $R_{\max}$ value of the Linpack benchmark. Each eServer system frame consists of 16 1.5 GHz Power5 processors. The nodes in the HPCx system are connected via IBM's High Performance Switch (HPS). It is located at the UK's CCLRC's Daresbury Laboratory, in England.

MareNostrum is a platform based on processors PowerPC, the architecture BladeCenter, a Linux system and a Myrinet interconnection. It has 31 racks dedicated to calculate. These racks have a total of 10,240 computational cores PowerPC 970 with a frequency of 2.3 GHz and 8 Gb of shared memory with 20 Tb of total memory. Each rack is formed by 6 Blade Centers. In total, each rack has a total of 336 processors and 672 Gb of memory. Each one has a rough peak performance of 3.1 T flops. The theoretical peak performance of this 40,960 cores system is about 65 T flops. It is located in the deconsecrated Chapel Torre Girona at the Polytechnic University of Catalonia, Barcelona in Spain.

Babel is a platform based on 10 racks (1024 nodes PowerPC 450, 20 Tb of memory). Four 850 MHz PowerPC 450 processors are integrated on each Blue Gene/P node. Each node can share 2 Gb of memory given a theoretical peak performance of 13.6 G flops (3.4 G flops per processor, 4 processor per node). The network is based on 3D Torus architecture (5.1 Gb/s; 3.5 μs latency) with a collective network (1.7 Gb/s; 2.5 μs latency). The theoretical peak performance of this 40,960 cores system is about 139 T flops. It is located at IDRIS (French national supercomputing service from CNRS) in Orsay, France.

4.1.1. Memory requirement

The amount of memory necessary for a simulation is dictated by the amount of 3D arrays necessary to solve the Navier–Stokes equations. In **Incompact3d**, there is up to 43 3D arrays in the parallel code, used for the computation of the velocity and pressure fields, for the computation of staggered FFTs (solving of the Poisson equation) and for the storage of the data for statistical analysis (Reynolds stresses, energy spectra, etc.) and post-processing visualisations. Even if it is more expensive in memory, that type of storage avoids penalizing computational efficiency by intensive disk accesses. Note also that this number of 3D arrays can be reduced when the flow configuration allows strong assumptions (statistical homogeneities or symmetries) to be made. The amount of memory A_m (in Gb) required for one simulation is given by

$$A_m \approx N_a S_d \left(\frac{n_x n_y n_z}{2^{30}} \right)$$

where N_a is the number of 3D arrays in the simulation and S_d the number of bytes for real representation (4 for a simulation performed in single precision and 8 in double precision). For example, a simulation with $2048 \times 1024 \times 1024$ mesh nodes requires ≈ 344 Gb of memory if the data are stored in single precision. Distributed over 512 computational cores, such a simulation will require ≈ 0.672 Gb per computational core. In practice, this requirement can be a limiting factor for platforms based on processors with limited memory per core (for instance, Babel provides only 0.5 Gb per core, which is not enough for the resolution taken here as an example).

4.2. Scalability and efficiency

In the context of High Performance Computing, there are two common notions of scalability. The first one is strong scaling, which is defined as how the restitution time of a simulation varies with the number of computational cores for a global fixed problem size: ideally, the problem will run twice as fast when the number of computational cores is doubled. The second one is weak scaling, which is defined as how the restitution time varies with the number of computational cores for a fixed problem size per computational core: ideally, when both the size of the problem and the number of computational cores are doubled, the restitution time remains constant. Note that in this paper, we mainly focused on strong scaling analysis (with associated speedup). In Computational Fluid Dynamics, most of the problems have a fixed global size and when the number of computational cores is increased, the simulation must perform better by a factor which justifies the additional resource employed. However, to have an accurate evaluation of the performance of the code, we are also presenting a weak scaling study (performed on HECToR). In order to measure the scalability and efficiency but also to test the good portability of the new version of **Incompact3d**, several simulations were performed with different number of mesh nodes and computational cores, depending on the memory capacity of each platform.

4.2.1. Strong scaling and efficiency

Ideally, the speedup should scale linearly with the number of computational cores in accordance with the potential computing

[3] This list is updated every six months and is available at http://top500.org.

[4] System based on Dell blade servers, Xeon dual core processors running at 2.66 GHz with 4 Gb of memory, theoretical peak performance of 18 T flops, sitting at position 369 on the mentioned list in June 2008.

[5] The XT4 system has been updated very recently. Each dual-core processor has been removed and replaced by a quad-core processor. This amounts to a total of 22,656 computational cores and the processor is an AMD 2.3 GHz Opteron. Each quad-core socket shares 8 Gb of memory. The theoretical peak performance of the system is now 208 T flops. Note that the scalability and efficiency of the code remains almost the same with this updated version of the XT4 system.

power which increases linearly. The parallel speedup factor S_f used in the present paper is defined by

$$S_f(c_{\text{used}}) = \frac{T(c_{\text{ref}})}{T(c_{\text{used}})}$$

where $T(c_{\text{ref}})$ is the elapsed time for one run with a number c_{ref} of computational cores, corresponding to the "reference simulation". As already mentioned, the memory available for one computational core is different on each platform. Basically, c_{ref} is the smallest number of computational cores required to run a simulation using the maximum memory available on each computational core. $T(c_{\text{used}})$ is the elapsed time for one run with a number c_{used} of computational cores.

The efficiency factor E_f used in this paper is defined by

$$E_f(c_{\text{used}}) = \frac{S_f(c_{\text{used}})}{c_{\text{used}}/c_{\text{ref}}}.$$

The parameters of the tests performed to check the performances of the code are presented in Table 1. To compute the scalability and the efficiency, we measure the computing time elapsed for one hundred time steps (with the same time step) for each run. Note also that these tests were performed with inflow/outflow in the x-direction and periodic in y and z directions (without a forcing term). These are indeed the boundary conditions we are using in our DNS of turbulent flows generated by fractal grids.

The results are presented in Figs. 3 and 4. As expected, the scalability and the efficiency of the code are quite good for all the simulations performed with a wide number of mesh nodes (from 4 million up to 4.3 billion). Indeed, despite the large amount of communication exchanges needed for the dual domain decomposition, the efficiency remains quite close to the ideal one for most of the tests performed up to 1024 computational cores but seems to be linked to the characteristics of each platform (network, processors, etc.). The smallest efficiency factor is found to be 0.69 for DNS_1 on Babel. Note that the network on Babel is optimum for simulations with more than one thousand computational cores and this platform seems not to be really suitable for most of our size configuration problems. The best performances are obtained on HECToR due to a very powerful network and high frequency processors. Also the relatively poor performances on Babel could be linked to the number of cores of each processor (quadri cores for Babel and dual cores for HECToR). We have also observed that when the number of mesh node n_x is too small, basically smaller than $\min(n_y, n_z)$, the scalability of the code becomes very poor. This is due to the large number of exchanges among the MPI-processes and the small size of these exchanges (directly linked to n_x). We did not try to solve this problem, as in our future simulations, the biggest size of the computational domain is in the x-direction.

4.2.2. Weak scaling

Weak scaling indicates how the restitution time varies with computational cores count with a fixed problem size per computational core. So in a weak scaling study when one doubles the number of computational cores one also doubles the global problem size. In Fig. 5, the weak scaling obtained on HECToR for the code is presented. The smallest problem size is 4,198,400 mesh nodes on 1 computational core, the largest one is 4,299,161,600 (almost 4.3 billion mesh nodes) on 1024 computational cores. It can be seen that the weak scaling is very good, the restitution time for one time step increasing from 6.16 s to only 7.8 s when going from 1 computational core to 1024. Even if the amount of communications is increasing when increasing the number of mesh nodes and the number of computational cores, the restitution time remains almost the same because all the communications are performed at the same time. The small differences observed are probably due to the latency of the system.

Note also that for simulation DNS_4 using 256 computational cores on HECToR, we have observed a performance of 0.92 G flops per computational core for a total of 253.52 G flops. There are many factors in computer performance other than raw floating-point computation speed, such as access performance, interprocessor communications, cache coherence and memory hierarchy, etc. For these reasons, a numerical code is only capable of a small fraction of the "theoretical peak" of a computational core. For simulation DNS_4, 16.44% of the "theoretical peak" of the computational core is reached, in good agreement with other fluid dynamics codes on massive parallel platforms (between 10% and 20% of the "theoretical peak" is usually reported).

The parallel strategy used for the code seems to be suitable for running massive DNS with up to 1024 computational cores (note that between 512 and 1024, the performance are linked to the characteristics of the platform). It is a clear improvement because it allows us to run our multiscale generated flow simulations with more than five billion mesh nodes which would obviously not be realistic on a current vector platform. In the next section, we consider in some more detail the communication cost associated with the present dual domain decomposition strategy.

4.3. Communication cost

In order to check the global performance of the parallel code, especially the cost of the dual domain decomposition strategy, several simulations have been conducted using Cray software (CrayPat and Cray Apprendice [23]) on HECToR. In order to take into account the communication cost into the efficiency of the code, the parallel efficiency E_p is also calculated in this section, using $E_p = 1 - t_{\text{comm}}/t_{\text{total}}$ where t_{comm} is the communication time.

Communication costs in MPI programs depend on both the number of block data exchanged among the MPI-processes and the size of each exchange. Naturally, they are linked to the number of computational cores (in this study, we have one MPI-process per computational core). Indeed, the number of exchanges increases with the number of MPI-processes whereas the size of each exchange decreases as the number of mesh nodes remains constant for a given simulation. Table 2 shows the communication cost for one simulation performed on HECToR with the same number of mesh nodes but with 4 different numbers of computational cores. As expected, due to the dual domain decomposition strategy, most of the time is spent to perform the different global transpose operations. Even if the communication cost increases from 24% for 16 computational cores up to 42% for 128, the scalability for this simulation remains close to the ideal one (see Fig. 4). As expected with the increase of the time spent in the communication, the parallel efficiency cannot remain acceptable when the number of computational cores is increased. However, this degradation is acceptable considering the good scaling of the global restitution time. As the number of mesh nodes remains constant, the global transpose operation concerns the same amount of data. The two main differences between the 4 simulations are the number and the size of the exchanges. It should also be important to note that for a given

Table 1
Parameters of the DNS for the scalability and efficiency evaluations.

DNS	(n_x, n_y, n_z)	Computational cores	Platform
DNS_1	(769, 1024, 1024)	64 → 512	HECToR
DNS_2	(257, 128, 128)	1 → 64	HECToR/HPCx/Babel
DNS_3	(1025, 2048, 2048)	512 → 1024	HECToR
DNS_4	(769, 512, 512)	16 → 256	MareNostrum/HPCx/Babel

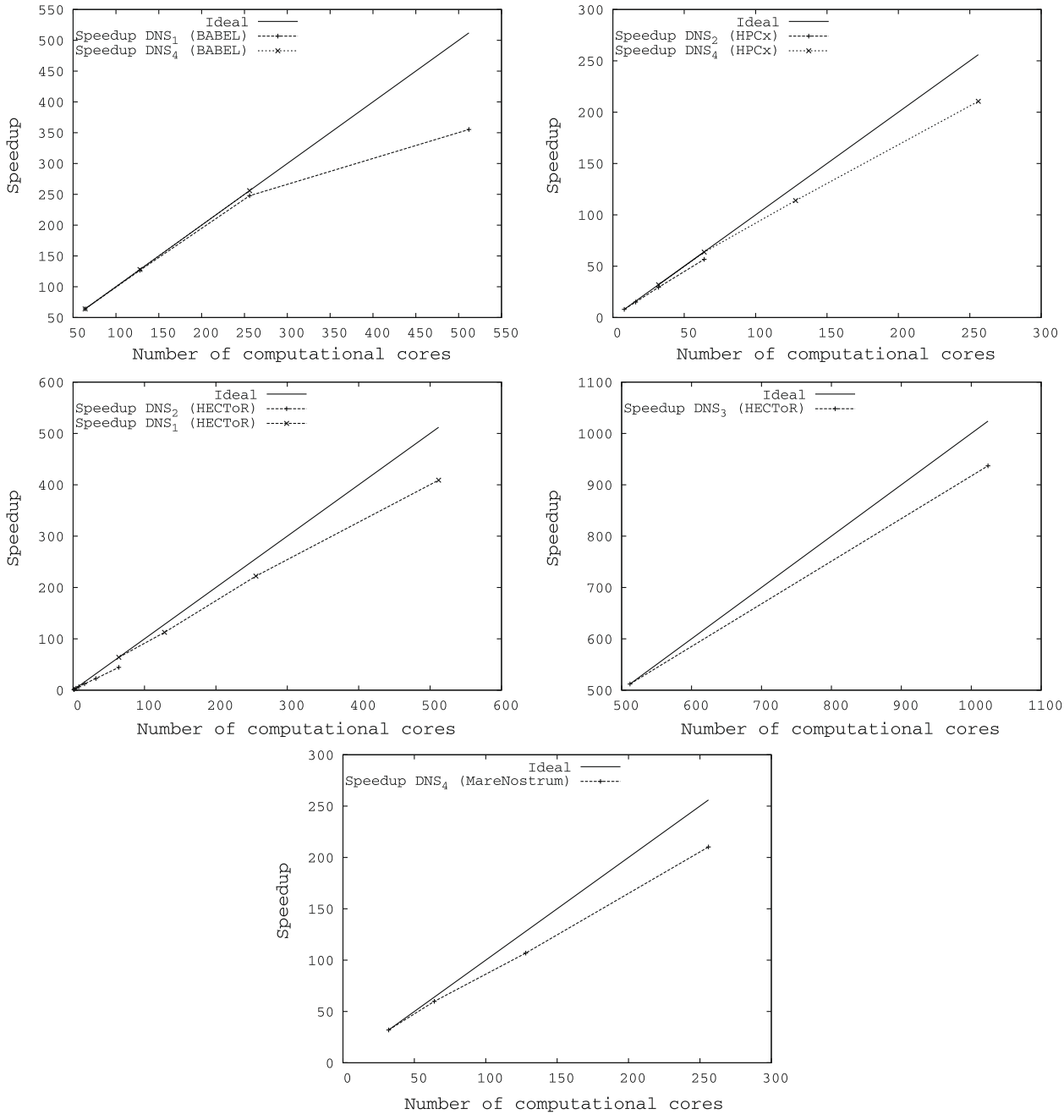

Fig. 3. Speedup plots of **Incompact3d** for various DNS on the four platforms considered here.

simulation, the size of each subdomain is the same because the number of mesh nodes is always a multiple of the number of computational cores. Furthermore, the relative cost of the Poisson equation remains nearly a constant of about 15% which means that the FFTW are quite efficient on this platform. The relative cost of the computation of the convective–diffusive terms is slowly decreasing when the number of computational core is increasing, maintaining a relatively correct efficiency. Indeed, as the time spent in the communication is increasing, it is logical to observe a decrease for the computation of the convective–diffusive terms. Note finally, that the total time (communications + Poisson + conv–diff) remains almost constant ($\approx 75\%$) when the number of computational cores is increasing.

One of our main challenge when we have developed the parallel version of the code, besides dealing with its complexity, communication and scaling, was to maintain the portability of the code by using open source library like MPI of FFTW. However, parallel platforms differ in a number of ways: (i) Processor performance: the processors can have very different performance levels in terms of the flop rates, caches sizes and memory bandwidths (see Babel and HECToR for instance), (ii) Networking bandwidth and latency: The bandwidth and latency costs for sending messages between processes can vary significantly, (iii) Network topology: The way in which the processors are connected can determine the relative costs of global communications and the optimal manner in which data exchanged should be performed. We had to consider the influence of these parameters on the dual domain decomposition and on the communication activity. For these reasons, the parallel version of the code was tested on a wide range of systems. Because the performances (scalability and efficiency) are quite good on each

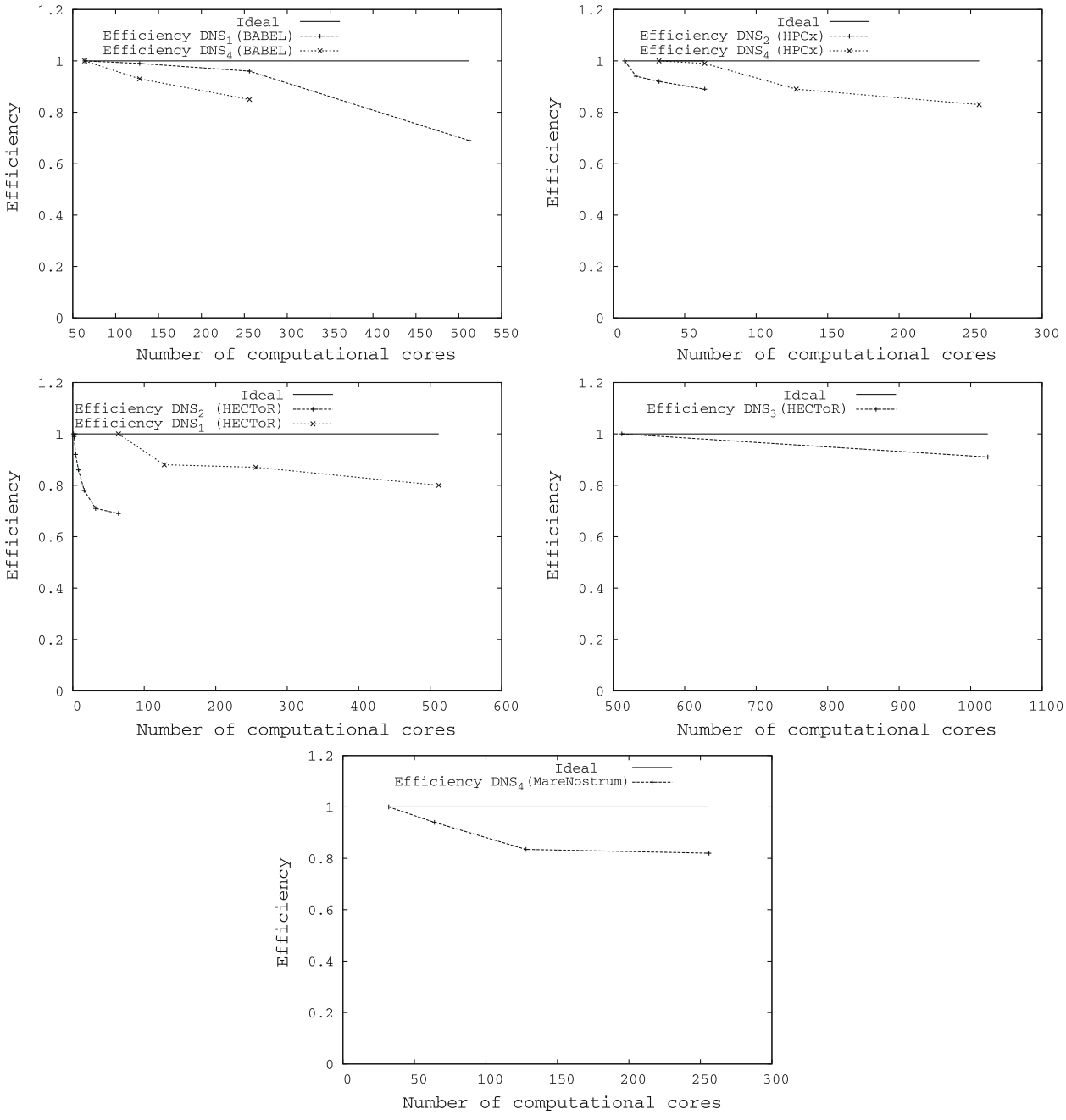

Fig. 4. Efficiency plots of **Incompact3d** for various DNS and platforms.

platform, we can say that all these parameters only have a relatively limited impact on the global performance of the code.

5. Results and comparisons with experiments

Results from two simulations of turbulent flow generated by fractal grids (cross and square grids, see Fig. 1) are presented and, in the case of the cross grid, compared with experimental data. An initial view on the vortex dynamics is presented using visualisations based on the streamwise velocity and the vorticity which provide an initial insight on how the different wakes, generated by different size bars of the grids, interact with each other and influence the behaviour of the flow further downstream. Results based on turbulent statistics are also presented.

One of the two simulations presented here corresponds to a fractal cross grid with three fractal iterations, the one of Fig. 3a of [7]. The second DNS corresponds to a square grid with three fractal iterations and has been performed with the aim of recovering one of the unusual results observed experimentally with fractal square grids: the intensity of the turbulence generated on the centreline builds up as the turbulence is convected downstream till a distance x_{peak} is reached from the grid where the turbulence intensity peaks and then a DHIT (Decaying Homogeneous Isotropic Turbulence) can be observed further downstream from that peak.

5.1. Configuration of the flows

The fractal cross grid consists of different sized crosses made of 2 rectangular bars and the fractal square grid consists of different sized squares with 4 rectangular bars. These fractal grids are completely characterised by the choice of pattern (cross or square) and

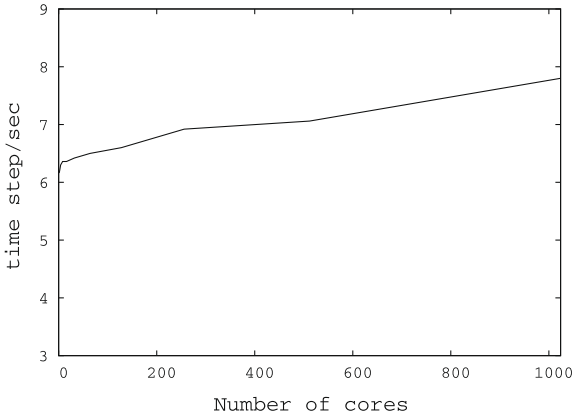

Fig. 5. Weak scaling of **Incompact3d** on HECToR.

Table 2
Time for communications, the computation of convection-diffusion terms and for the Poisson equation's solution cost for different simulations with the same number of mesh nodes but with different numbers of computational cores.

Cores	Comm. (%)	Poisson (%)	Conv–diff (%)	Global time (s)	Efficiency E_f	Parallel efficiency
16	24.32	15.81	29.4	10,062	1	0.757
32	28.28	15.18	27.1	5121	0.98	0.717
64	33.87	14.19	22.9	2861	0.88	0.661
128	42.33	12.99	19.8	1412	0.89	0.577

- the number of fractal iterations N
- the bar lengths $L_j = R_L^j L_0$ and thicknesses $t_j = R_t^j t_0$ (in the plane of the grid, normal to the mean flow) at iteration $j, j = 0, \ldots, N-1$
- the number B^j of patterns at iteration j.

Here, $B = 4, N = 3, R_L = 1/2$ and $R_t \leqslant 1$ for the two grids. By definition, $L_0 = L_{\max}, L_{N-1} = L_{\min}, t_0 = t_{\max}$ and $t_{N-1} = t_{\min}$. The blockage ratio σ of these grids is the ratio of their total area to the area T^2 of the tunnel's cross section. It is also of interest to define the thickness ration $t_r = t_{\max}/t_{\min}$. Unlike regular grids, these fractal grids do not have a well-defined mesh size, and [7] have therefore introduced an effective mesh size for fractal grids defined as follows

$$M_{\text{eff}} = \frac{4T^2}{P}\sqrt{1-\sigma}$$

where P is the fractal grid's perimeter length. Note that the fractality of the grids influences M_{eff} via the perimeter P which can be extremely long in spite of being constrained to fit within the area T^2. When applied to regular grid, this definition of M_{eff} gives the correct mesh size M of the grid. The experimental parameters defining each grid are given in Table 3 in metric units for an overview of the experimental set-up of [7,19]. This table also includes U_∞, the velocity at the entrance of the wind tunnel. Note also that the grids have the same 5 mm thickness in the direction of the mean flow (Fig. 6).

Table 3
Experimental parameters of the wind tunnel measurements performed by [7] and grid parameters for the two simulations.

Grid	t_r	$t_{\max}$ (mm)	$L_{\max}$ (mm)	σ (%)	M_{eff} (mm)	U_∞	N
Cross (Expe)	3.3	62	910	40	114	12 m/s	3
Cross (DNS_1)	3.3	62	910	40	114		3
Square (Expe)	3	10.7	230	20	26	10 m/s	4
Square (DNS_2)	3	6.3	115	20	26		3

DNS_1 corresponds to the simulation of the turbulent flow generated by the fractal cross grid and DNS_2 to the turbulent flow generated by the fractal square grid. The numerical parameters for these simulations are determined to be those of [7]. The lateral lengths of the computational domain L_y and L_z are equivalent to the cross-sectional size of the wind tunnel T and the computational size in the streamwise direction is $L_x = 2T$. Note that for DNS_1, we have $L_x = 25.3M_{\text{eff}}$ and for $DNS_2, L_x = 18.6M_{\text{eff}}$. The targeted grids are determined to be as close as possible to the experimental ones in [7], except for the Reynolds number (based on U_∞ and M_{eff}), and, in the case of DNS_2, for the number of fractal iterations (3 instead of 4). Indeed, despite the use of high resolution DNS (up to 382 millions mesh nodes), $Re_{M_{\text{eff}}} = \frac{U_\infty M_{\text{eff}}}{\nu}$ must be reduced from 87,400 to 1860 for the cross grid and from 20,800 to 4430 for the square grid. Consequently, only a qualitative agreement with experiments can be expected for these simulations. The numerical parameters of each simulation can be found in Table 4.

Due to the multiscale nature of these grids, the number of mesh nodes is of crucial importance because all the scales need to be accurately represented. A preliminary study [11] has shown that five mesh nodes is enough to discretize the smallest thickness of a fractal grid for the Reynolds numbers considered here. In this previous work, the collection time to obtain the statistical data was 15 M_{eff}/U_∞. In the present study, statistics are obtained by averaging over a collection time of 30 M_{eff}/U_∞ but also by using the symmetries of the flow in order to have well converged statistics. This way, we obtain mean flow and turbulence profiles as functions of streamwise distance x/M_{eff} or lateral coordinate y/T.

5.2. Turbulence generated by a fractal cross grid (DNS_1)

A first illustration of the flow is provided in terms of an isosurface of the instantaneous modulus of the vorticity vector in Fig. 7 where a fully turbulent flow can be observed. In order to make some comparisons with the experimental data, we also plot statistics of the turbulent flow. In Fig. 8, we compare the streamwise evolution of the mean flow velocity U along the centreline ($y/T = z/T = 0$) for the experimental and numerical data. In their experimental measurements, Hurst and Vassilicos [7] observed that fractal cross grids generate non-zero values of $\frac{\partial U}{\partial x}$ at distances x/M_{eff} from the grid on the tunnel's centreline where regular grids do not. This behaviour is also observed in our simulation which therefore agrees qualitatively with the wind tunnel experiments at the large enough streamwise distances where the hot wire measurements were taken (see Fig. 8, left). It should be stressed again that the Reynolds number in the experiment is much higher than in our simulation and therefore no exact quantitative agreement should be expected. It is nevertheless interesting that our simulation reveals a significantly large recirculation region just behind the bigger cross, as identified by the visualisation of the isosurface where the mean flow velocity U is equal to zero (Fig. 8, right). Note also that smaller recirculation regions can be found behind the bars of the crosses corresponding to the second fractal iteration. The large recirculation region near the centreline is also reflected in the negative values of the mean flow velocity at $x/M_{\text{eff}} \leqslant 7$ on the centreline. These first conclusions represent a clear prediction for future experimental measurements closer to the grid and at lower Reynolds numbers. It would be useful to also study in the future the influence of the streamwise thickness of the cross grid but also the interaction between the bigger vertical and horizontal wakes and its effects on the length and on the shape of this large recirculation region.

As mentioned by [7], these gradients of U along x must be offset by lateral gradients of U along the y and z axis. Fig. 9 (left) shows the mean flow evolution profile in the lateral y-direction and establishes that there is a qualitative agreement with the

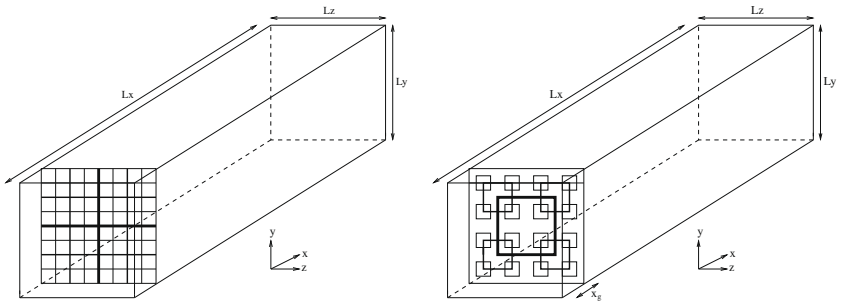

Fig. 6. Schematic view of the flow configuration for the fractal cross grid (DNS_1, left) and the fractal square grid (DNS_2, right).

Table 4
Numerical parameters of the two simulations.

DNS	Domain ($\times M_{\text{eff}}$)	Mesh nodes	$Re_{M_{\text{eff}}}$	Δt
DNS_1	$25.3 \times 8.43 \times 8.43$	$769 \times 256 \times 256$	1820	$1.2310^{-3} M_{\text{eff}}/U_\infty$
DNS_2	$18.6 \times 9.3 \times 9.3$	$1152 \times 576 \times 576$	4430	$6.0510^{-4} M_{\text{eff}}/U_\infty$

experimental and numerical data, even though they are taken at different distances from the grid and different Reynolds numbers.

In Fig. 10, we plot the centreline streamwise decay of turbulence intensities. Our numerical simulation confirms the high turbulence intensities returned by fractal turbulence actuation. Whilst the streamwise decay is also qualitatively similar between experiment and simulation, the actual turbulence intensity values are found to be different close to the grid. The large recirculation bubble identified in the DNS could be responsible for the higher values observed in the numerical data. Indeed, because the Reynolds numbers are significantly different between our simulation and the experimental set-up, it may well be possible that the recirculation bubble observed in our simulation diminishes in size as the Reynolds number increases. Nevertheless, at distances $x/M_{\text{eff}} = 20$ and beyond, the turbulence intensities are found to be quantitatively similar.

Finally, in Fig. 9 (right) we compare the large-scale isotropy indicator u'/v' obtained experimentally and numerically far downstream from the grid. Even through our DNS statistics are not as well converged as the wind tunnel statistics, one can make out a general agreement. A similar shape for the profile of the ratio u'/v' is found in experiments and DNS with reduced values in the centre of the domain. Two clear peaks can be identified for $y/T = \pm 0.25$ with a $u'/v' = 1.5$ in the numerical data. Even if there is no experimental data near the wall of the wind tunnel, the maximum values for u'/v' are located close to $y/T = \pm 0.35$, which is not so far from the numerical location but not exactly at the same place either. Note that even though this comparison is made at a

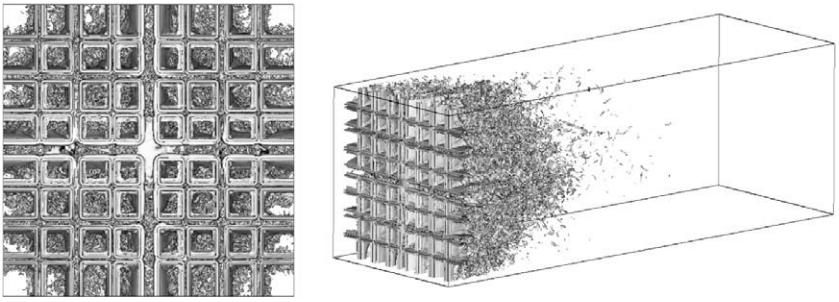

Fig. 7. Isosurface of the instantaneous modulus of the vorticity vector for the turbulent flow generated by a fractal cross grid. The isopycnal value is $2U_\infty/M_{\text{eff}}$.

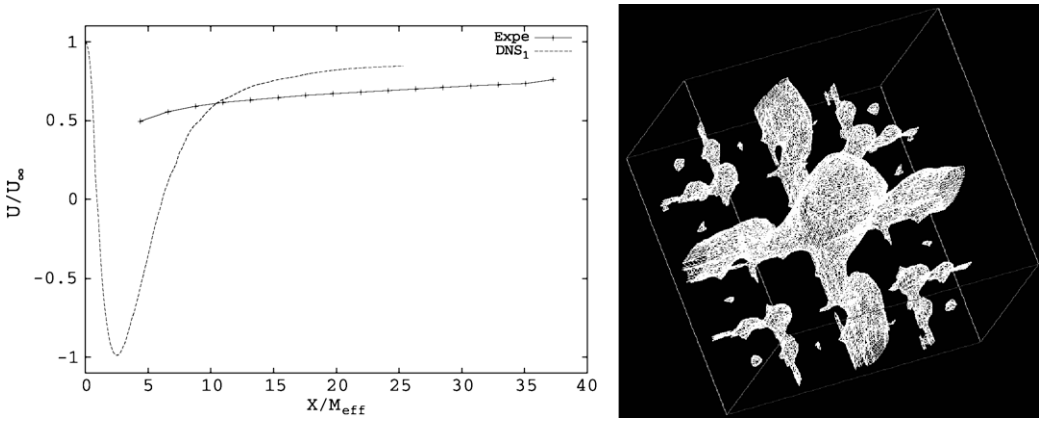

Fig. 8. Streamwise evolution of U/U_∞ on the centreline for the experimental data and the numerical data (left). Isosurface where the mean flow velocity U is equal to zero for the numerical data (right).

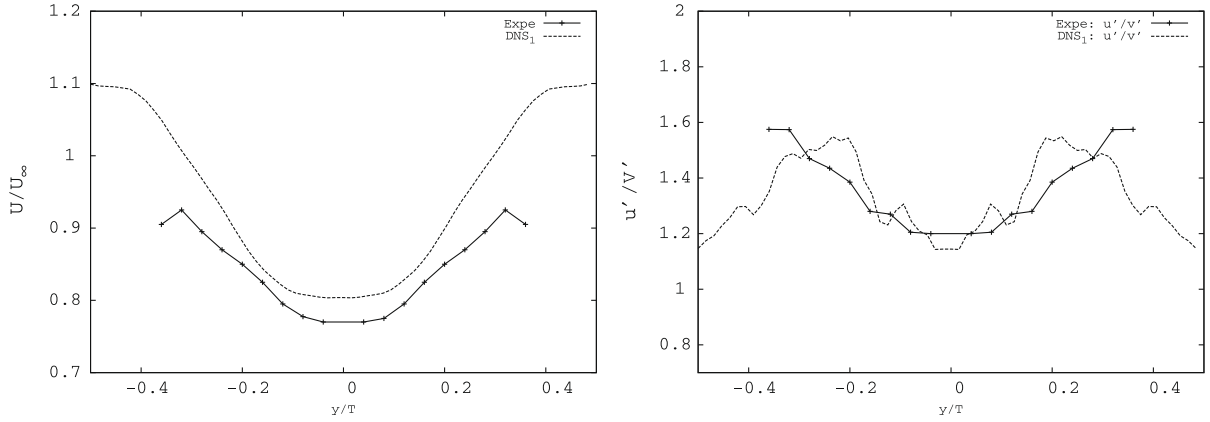

Fig. 9. Lateral evolution of the mean streamwise velocity (left) and lateral evolution of the large-scale isotropy indicator u'/v' (right): experimental data at $x/M_{\text{eff}} = 38$ and numerical data at $x/M_{\text{eff}} = 20$.

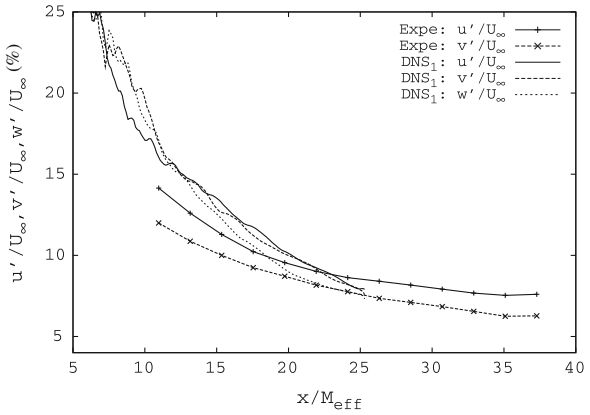

Fig. 10. Turbulence decay on the centreline for experimental data and numerical data.

value of x/M_{eff} large enough to expect moderate Reynolds number effects on the mean flow, there could still be significant Reynolds effect on u'/v'. Finally, it is important to notice that the lateral confinement in the wind tunnel that is not taken into account in present DNS due to the use of periodic boundary conditions in the lateral directions could also be responsible for the differences observed in this section. However, we do not think that these differences are due the boundary layers developing on the lateral walls of the wind tunnel because measurements have shown that their thicknesses are very thin. The main effect is probably related to the blocking effect introduced by the walls in the normal direction. This kinematic effect could be taken into account in DNS without having to compute the boundary layers (that would be very expensive) through the use of free-slip boundary conditions. This test will be carried out in a future study in order to better distinguish these blocking effects from the Reynolds number influence.

5.3. Turbulence generated by a fractal square grid (DNS_2)

Instantaneous streamwise velocity visualisations are presented in Figs. 11 and 12. A non-homogeneous turbulent field is obtained in our simulation close to the grid with the clear presence of wakes of three different sizes, corresponding to the three fractal iterations on the grid. Hurst and Vassilicos [7], in their experimental measurements, observed a production of turbulence near the centreline ($y/T = 0$). The visualisations suggest sequential interactions between wakes from small-scale ones all the way up to the larger scale ones. These interactions could be responsible for the streamwise production of turbulence, as observed by [7]. However, the computational domain in the streamwise direction is not long enough to observe a fully homogeneous turbulence as in the experimental measurements because the sequential wake interactions mechanism is not yet over. Fig. 12 shows that the imprint of the fractal square grid survives far downstream from the grid and strongly influences the flow. Future simulations will need to have an increased computational size in the streamwise direction and also higher values of the thickness ratio t_r. Indeed, Hurst and Vassilicos [7] observed that the turbulence starts decaying closer to the fractal grid for higher t_r values.

In their wind tunnel experiments, Hurst and Vassilicos [7] noticed that, on the centreline, the turbulence intensities build up downstream from the grid till a point is reached where they peak and then decay further downstream. Even if it is not possible to observe an extended decay region because of the relatively small size of the computational domain in the streamwise direction, one can notice an increase of u' near the centreline (Fig. 13, right), in agreement with the instantaneous visualisations of the streamwise velocity, probably resulting from the sequential interactions between the wakes seen in Fig. 12. In Fig. 13, we notice a small acceleration of the mean flow velocity on the centreline of the grid whereas behind the big bars ($y/T = 0.03$) recirculation bubbles seems to appear. Note that in their wind tunnel experiments, Hurst and Vassilicos [7] also found values higher than 1 for U/U_∞ on the centreline. Future investigations will need to address the effects of the number of fractal iterations on the mean flow velocity close to the grid.

Some of these first observations can be investigated a bit further in terms of power spectra of the streamwise velocity (Fig. 14). These power spectra, obtained for $z/T = 0$ at 3 different locations in the y-direction ($y/T = 0$, $y/T = 0.06$ and $y/T = 0.03$) and 3 different locations in the streamwise direction, are estimated using the periodogram technique [17]. Eleven sequences are used here, corresponding to a global time duration of $T = 28.35M_{\text{eff}}/U_\infty$ overlapped at 50%, with the use of a Hanning window. A spectrum with a very approximate $-5/3$ scaling region may be appearing as early as $x/M_{\text{eff}} = 3.5$ behind the big square ($y/T = 0.03$) whereas on the centreline ($y/T = 0$) such a $-5/3$ scaling region does not appear in the spectrum all the way to the end of the computational domain. Furthermore, the energy spectra clearly show the non-homogeneous character of this flow close to the grid. This spectral analysis suggests that the flow is almost homogeneous at the end of the computational domain. The evolutions of the energy spectra confirm the evolutions of the turbulence intensities. A more detailed analysis of the evolution of the energy spectra in the regions where

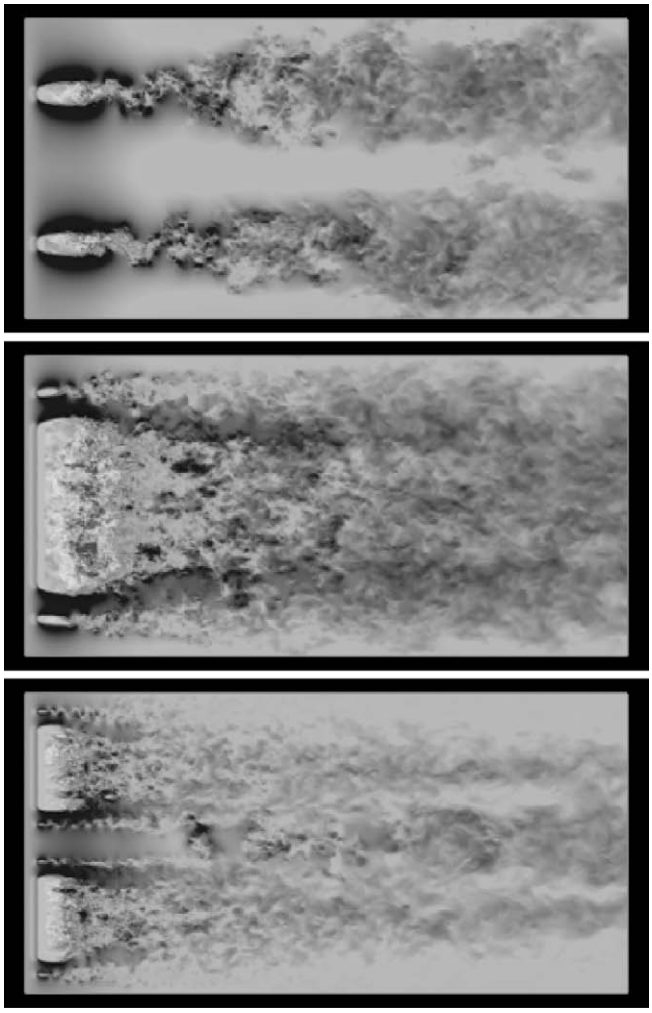

Fig. 11. Instantaneous streamwise velocity in the $(x-z, y/T=0)$ plane (top), in the $(x-z, y/T=0.06)$ plane (middle) and in the $(x-z, y/T=0.06)$ plane (bottom). The black color corresponds to 1.5 and the white one to -1.5.

we can observe a clear production of turbulence could help to understand how the turbulence builds up from the small scales to the big scales and perhaps also provide insights on the non-Kolmogorov aspects of the decay region of this flow [7,19].

6. Conclusion and perspectives

A numerical strategy suited for massive DNS of incompressible flows is presented in this paper. The approach proposed is based on a combination of high-order compact schemes, immersed boundary method and a dual domain decomposition. A priori, the combination of high-order schemes with domain decomposition can be problematic because of the implicit nature of the schemes. However, the parallelisation of **Incompact3d** has been successfully performed and the code is able to solve computationally very large fluid flow problems with a good efficiency on a large range of massive parallel platforms. Portability has been successfully realised using generic independent-platform tools and protocols like MPI and FFTW. The global transformation operations are quite expensive with up to 40% of the total cost of a simulation. However, we have shown that the scalability remains acceptable up to 1024 computational cores (between 512 and 1024 cores depending on the platform considered).

The high performance computing landscape is set to change to the point that it would be beneficial to improve the current parallel code. Indeed, the dual domain strategy does not seem to be efficient enough when thousands of processors are used. The main limitation of the parallel code is that it can only use a limited number of computational cores n_c ($n_c < min(n_y, n_z)$), the main consequences being a memory requirement problem and a restitution time longer than desirable for production purposes. It is absolutely clear to us that the efficient and cost-effective exploitation of current and future high-performance parallel platforms is an essential element for our research and the research of all those at the forefront of fluid mechanics in general and turbulent flows in particular.

Following a strategy used by CFD codes based on spectral methods which are currently running on thousands of computational cores, a new strategy based on a "pencils" domain decomposition

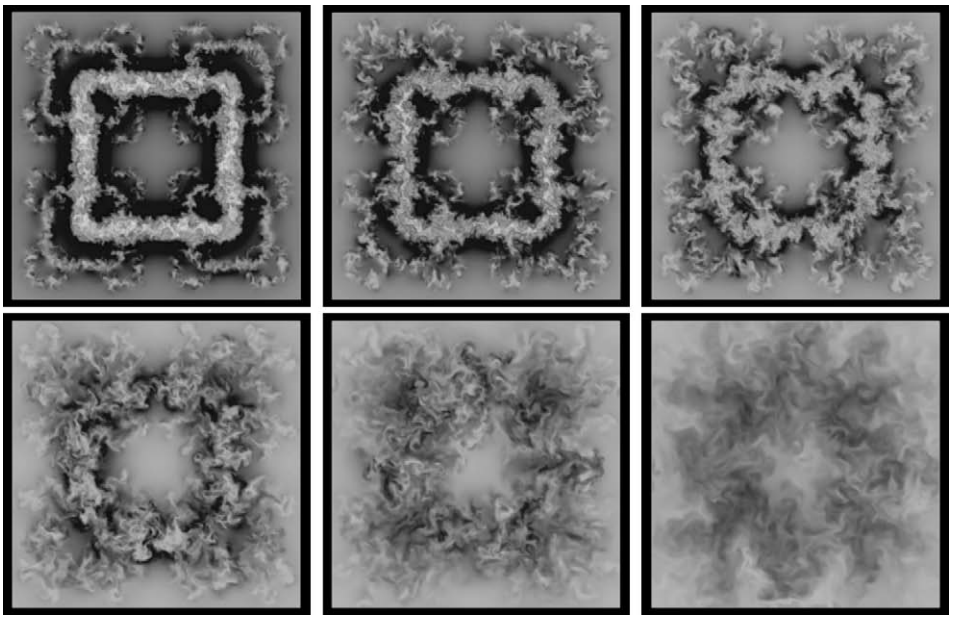

Fig. 12. Instantaneous streamwise velocity in the $(y-z, x/M_{\text{eff}} = 1.5, 2.5, 3.25, 5, 8 \text{ and } 16)$ planes. The black color corresponds to 1.5 and the white one to -1.5.

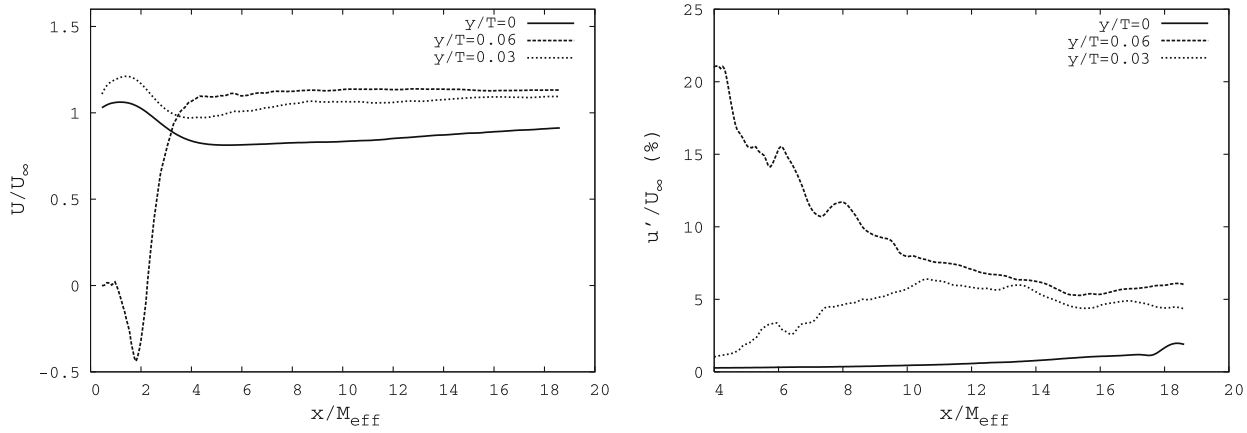

Fig. 13. Mean flow profile U/U_∞ in the streamwise direction for $z = L_z/2$ and for different value of y (left). Evolution of u'/U_∞ in the streamwise direction for $z = L_z/2$ and for different value of y (right).

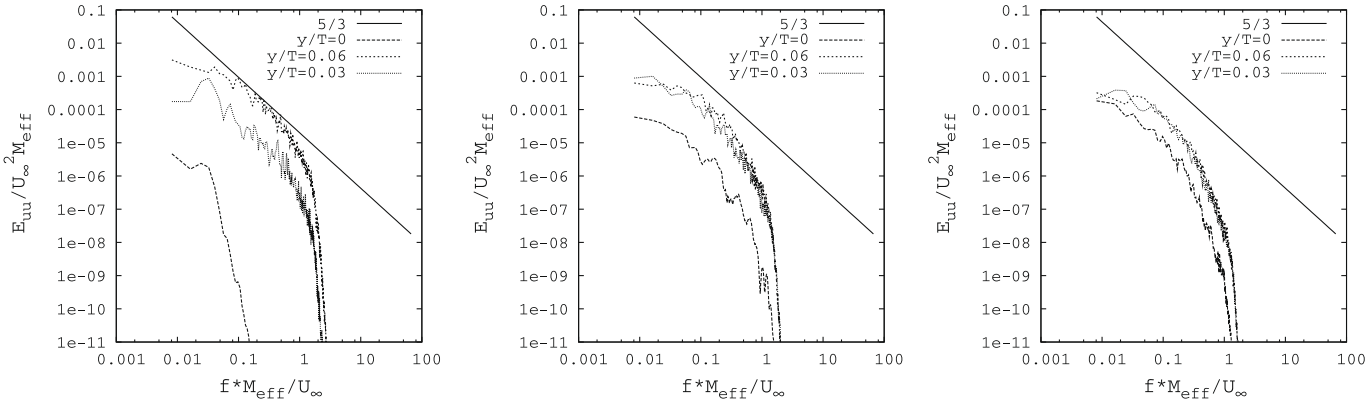

Fig. 14. Power spectra of the streamwise velocity at three different locations in y, for $z/T = 0$ and for $x/M_{\text{eff}} = 3.5$ (left), $x/M_{\text{eff}} = 7$ (middle) and $x/M_{\text{eff}} = 14$ (right).

could be considered in order to run bigger simulations and/or drastically reduce the wall clock time of our current typical size simulations. This strategy will be of course fully compatible with our implicit schemes in space. It seems to be a relatively expensive strategy in terms of communication as many global transpose operations are required. However, [4] have made a comparison between a dual and a triple 2D domain decomposition (based on "pencils") for their spectral code, and found the triple 2D decomposition to be only 10% more expensive in time for the same simulation, same number of mesh nodes and same number of processors. This is clearly a very small price to pay for the advantage of being able to significantly increase the number of processors and run much bigger simulations and/or drastically reduce the wall clock time of our DNS.

Physically, the results of DNS of fractal generated turbulence are encouraging and some good qualitative agreement with experimental data has been found despite the difference in Reynolds number. We have been able to reproduce the streamwise production of turbulence downstream the fractal square grid, first observed in the wind tunnel measurements [7]. Our fractal square DNS has also offered some initial insights on the cause of the streamwise turbulence production. Future DNS studies will need to increase the size of the computational domain in the streamwise direction in order to catch the decay region which follows the turbulence production region. Future simulations of multiscale-generated turbulent flows are also important because of many technological applications in mixing, combustion, aeroacoustic, etc.

Acknowledgements

Calculations were carried out in the United Kingdom, France and Spain on 4 different platforms. The "French" part of this work was granted access to the HPC resources of IDRIS under the allocation 2008-0210912 made by GENCI (Grand Equipement National de Calcul Intensif) in collaboration with the "Laboratoire d'Etudes Aérodynamiques" in the University of Poitiers. The "Spain" part of this work was carried out under the HPC-EUROPA project (RII3-CT-2003-506079), with the support of the European Community – Research Infrastructure Action under the FP6 "Structuring the European Research Area" Program). The "UK" part of this work was carried out thanks to the EPSRC UK Turbulence consortium (EPSRC Grant EP/D044073) and EPSRC Grant EP/F051468 which made CPU time available to us on HECToR.

The authors are grateful to Roderick Johnstone, Martyn Foster and Vincent Hurtevent for help with the parallel code and for the correct use of each platform. They also thank Dr. Ning Li for his precious comments in the corrected paper. They also acknowledge support from EPSRC Research Grant EP/E00847X.

References

[1] Avital EJ, Sandham ND, Luo KH. Stretched Cartesian grids for solution of the incompressible Navier–Stokes equations. Int J Numer Methods Fluids 2000;33:897–918.

[2] Cain AB, Ferziger JH, Reynolds WC. Discrete orthogonal function expansions for non-uniform grids using the fast Fourier transform. J Comp Phys 1984;56:272–86.

[3] Deville MO, Fischer PF, Mund EH. High-order methods for incompressible fluid flow. Cambridge University press; 2002.
[4] Donzis DA, Yeung PK, Pekurovsky D. Turbulence simulations on $o(10^4)$ processors. Tera Grid conference; 2008. Available from: http://www.sdsc.edu/us/resources/p3dfft/.
[5] Gropp W, Lusk E, Skjellum A. Using MPI: portable parallel programming with the message passing interface. Cambridge: MIT Press; 1994.
[6] Hoyas S, Jimenez J. Scaling of the velocity fluctuations in turbulent channels up to Re = 2003. Phys Fluids 2006;18:011702.
[7] Hurst D, Vassilicos JC. Scalings and decay of fractal-generated turbulence. Phys Fluids 2007;19:035103.
[8] Kim J, Moin P. Application of a fractional-step method to incompressible Navier–Stokes equations. J Comp Phys 1985;59:308–23.
[9] Laizet S, Lamballais E. Direct-numerical simulation of the splitting-plate downstream-shape influence upon a mixing layer. CR Mecanique 2006;334:454–60.
[10] Laizet S, Lamballais E. High-order compact schemes for incompressible flows: a simple and efficient method with the quasi-spectral accuracy. J Comp Phys 2009;228(16):5989–6015.
[11] Laizet S, Vassilicos JC. Multiscale generation of turbulence. J Multiscale Modelling 2009;1:177–92.
[12] Lamballais E, Silvestrini J, Laizet S. Direct Numerical Simulation of a separation bubble on a rounded finite-width leading edge. Int J Heat Fluid Flow 2008;29:612–25.
[13] Laurendeau E, Jordan P, Bonnet JP, Delville J, Parnaudeau P, Lamballais E. Subsonic jet noise reduction by fluidic control: the interaction region and the global effect. Phys Fluids 2008;20(10):101519.
[14] Lele SK. Compact finite difference schemes with spectral-like resolution. J Comp Phys 1992;103:16–42.
[15] Parnaudeau P, Carlier J, Heitz D, Lamballais E. Experimental and numerical studies of the flow over a circular cylinder at Reynolds number 3900. Phys Fluids 2008;20:085101.
[16] Parnaudeau P, Lamballais E, Heitz D, Silvestrini JH. Combination of the immersed boundary method with compact schemes for DNS of flows in complex geometry. In Proc DLES-5, Munich; 2003.
[17] Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical recipes. Cambridge: Cambridge University Press; 1992.
[18] Sandham ND, Howard RJA. DNS of turbulence using massively parallel computers. In: Emerson DR, Ecer A, Periaux J, Satofuka N, Fox P, editors. Parallel computational fluid dynamic. Elsevier Science B.V.; 1998. p. 23–32.
[19] Seoud RE, Vassilicos JC. Dissipation and decay of fractal-generated turbulence. Phys Fluids 2007;19:105108.
[20] Swarztrauber PN. The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle. SIAM Rev 1977;19:490–501.
[21] Wilhelmson RB, Ericksen JH. Direct solutions for Poisson's equation in three dimensions. J Comput Phys 1977;25:319–31.
[22] Mitsuo Yokokawa, Ken'ichi Itakura, Atsuya Uno, Takashi Ishihara, Yukio Kaneda. 16.4-T flops Direct Numerical Simulation of turbulence by a Fourier spectral method on the Earth Simulator. In: Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing, Los Alamitos, CA, USA: IEEE Computer Society Press; 2002. p. 1–17.
[23] Available from: http://docs.cray.com/books/S-2376-41/S-2376-41.pdf.
[24] Available from: http://www.fftw.org.