AIMETA '01
XV Congresso AIMETA di Meccanica Teorica ed Applicata
15th AIMETA Congress of Theoretical and Applied Mechanics

A 4TH-ORDER-ACCURATE, PARALLEL NUMERICAL METHOD FOR THE DIRECT SIMULATION OF TURBULENCE IN RECTANGULAR AND CYLINDRICAL GEOMETRIES

M. QUADRIO[1], P. LUCHINI[2]

[1] Dipartimento di Ingegneria Aerospaziale Politecnico di Milano
[2] Dipartimento di Ingegneria Meccanica Università di Salerno

SOMMARIO

Si presenta un metodo efficiente per la soluzione numerica diretta delle equazioni di Navier–Stokes per fluido incomprimibile, in geometrie cartesiana e cilindrica. Il metodo è basato su schemi a differenze finite compatti del IV ordine per la direzione normale alle pareti, e su una discretizzazione di Fourier nelle due direzioni omogenee, ed è in grado di sfruttare la potenza di calcolo offerta a basso costo da cluster di Personal Computers multiprocessore. Anche nella formulazione cilindrica si utilizza l'efficiente formulazione in velocità–vorticità, risolvendo inoltre in modo innovativo il problema della variazione della risoluzione in direzione azimutale con il raggio.

ABSTRACT

An efficient numerical method for the direct numerical simulation of the incompressible Navier–Stokes equations in rectangular and cylidrical geometries is presented. The method is based on Fourier expansions in the homogeneous directions and fourth-order accurate, compact finite difference schemes over a variable-spacing mesh in the wall-normal direction. It can take advantage of parallel computing both on shared memory machines and/or distributed memory systems, and is designed towards the use with a cluster of commodity Personal Computers for low-cost, high-performance computing. The cartesian version of the method solves two scalar, independent equations for the wall-normal components of velocity and vorticity, for best computational efficiency. The innovative extension of the numerical method to cylindrical geometries manages to exploit a similar computationally efficient scheme. The typical disadvantage of polar coordinate system, of a varying azimuthal resolution in the radial direction is solved by adopting a Fourier expansion of the flow variables where the number of azimuthal modes varies with the radial coordinate.

1. INTRODUCTION

The direct numerical simulation (DNS) of the Navier–Stokes equations for incompressible fluids in geometrically simple, low-Reynolds number turbulent wall flows has become in the last years a valuable tool for turbulence research [3]. The relevance of such flows is enormous, from the point of view of practical interest and basic research, and a number of studies exists concerning simple flows in cartesian coordinates. Flows which can be easily described in cylindrical coordinates (turbulent flows in pipes and annular ducts) are by no means less interesting, but, despite their practical relevance, have not been studied so deeply through DNS. This can be at least partially ascribed to the numerical difficulties associated with the cylindrical coordinate system.

For the cartesian coordinate system, a very effective formulation of the equations of motion was presented in [1], and since then employed in many of the DNSs of turbulent wall flows in planar geometries. It consists in the substitution of the continuity and momentum equations with two scalar equations, one (second-order) for the normal component of vorticity and one (fourth-order) for

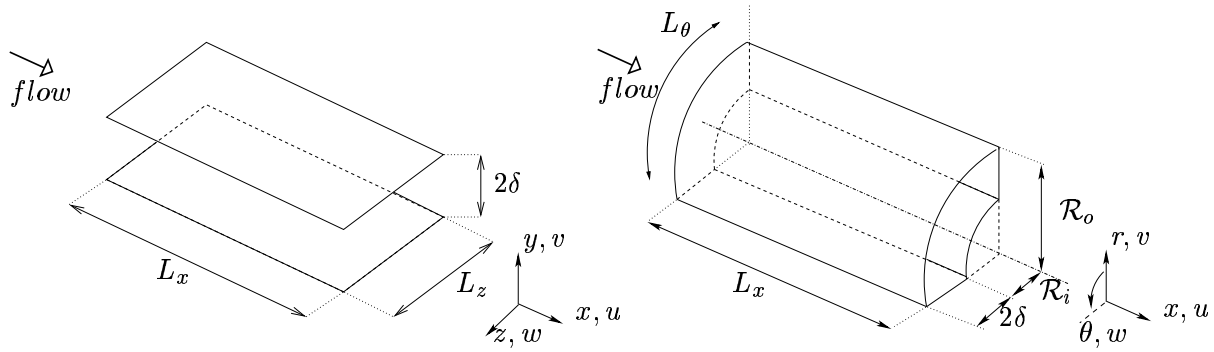

Figure 1: Sketch of the computational domain for the cartesian (left) and cylindrical (right) coordinate system

the normal component of velocity. This procedure is appealing, since pressure is removed from the equations, and the recovery of the other two velocity components can be done through the solution of a 2x2 *algebraic* system (a very cheap procedure from a computational point of view), when a Fourier expansion is adopted for the homogeneous directions.

An extension of the efficient cartesian formulation is still missing for the cylindrical case. Most of the existing numerical studies of turbulent flow in cylindrical coordinates write the governing equations in primitive variables, and use each a different numerical method: they range from second-order finite difference schemes (for example in [7]) to finite volumes to complex spectral multi-domain techniques (as in [2]), but most often remain inside the pressure-correction approach. The work by [6] is based on a spectral discretization, but calculation of pressure is still needed for the numerical solution of the equations. The use of cylindrical coordinates is moreover particularly hampered by the unwanted increase of azimuthal resolution of the computational domain with decreasing radial coordinate.

2. CARTESIAN COORDINATES: THE GOVERNING EQUATIONS

The cartesian coordinate system used in the present work is illustrated in figure 1 (left). The reference length δ is taken to be one half of the channel height. The reference velocity is chosen to be the velocity U_P of the laminar Poiseuille flow with the same mass flux, so that a Reynolds number can be defined as $\mathsf{Re} = \frac{U_P \delta}{\nu}$, where ν is the kinematic viscosity of the fluid.

Following for example [1], the non-dimensional Navier–Stokes equations for an incompressible fluid in cartesian coordinates, assumed to be periodic in both in the streamwise and spanwise direction, can be conveniently Fourier-transformed along the x and z coordinates. The wall-normal component η of the vorticity vector, after transforming in Fourier space, is by definition given by:

$$\hat{\eta} = i\beta\hat{u} - i\alpha\hat{w} \tag{1}$$

where the hat indicates the Fouries component of the variable, and the symbols α and β respectively denote the streamwise and spanwise wavenumbers. A one-dimensional differential second-order evolutive equation for $\hat{\eta}$ which does not involve pressure can be written by taking the radial component of the curl of the momentum equation, obtaining:

$$\frac{\partial \hat{\eta}}{\partial t} = \frac{1}{\mathsf{Re}} \left(D_2(\hat{\eta}) - k^2 \hat{\eta} \right) + i\beta H_{\hat{u}} - i\alpha H_{\hat{w}} \tag{2}$$

In this equation, D_2 denotes the second derivative operator in the wall-normal direction, $k^2 = \alpha^2 + \beta^2$, and the nonlinear terms are grouped in the definitions of $H_{\hat{u}}$, $H_{\hat{v}}$ and $H_{\hat{w}}$.

The two boundary conditions needed for the solution of (2) are $\hat{\eta} = 0$ at the channel walls.

An equation for the wall-normal velocity component $\hat{v}$ can be written by using the continuity equation transformed in Fourier space. Elimination of pressure can be achieved by proper manipulation

of the momentum equations, eventually obtaining the following fourth-order, scalar equation:

$$\frac{\partial}{\partial t}\left(D(\hat{v})-k^2\hat{v}\right)=\frac{1}{\mathsf{Re}}\left(D_4(\hat{v})-2k^2D_2(\hat{v})+k^4\hat{v}\right)-k^2H_{\hat{v}}-D\left(i\alpha H_{\hat{u}}+i\beta H_{\hat{w}}\right) \tag{3}$$

The four boundary conditions needed for the solution of (3) are $\hat{v}=0$ and $\partial\hat{v}/\partial y=0$ at the walls.

3. CARTESIAN COORDINATES: THE NUMERICAL METHOD

Both equations (2) and (3) have the structure of a forced Stokes equation, where the forcing term is represented by derivatives of the nonlinear terms. The numerical evaluation of the forcing term can be efficiently done if the Fourier components of the velocity are transformed back in physical space, the nonlinear terms evaluated by simple products and then retransformed into the Fourier space, where their derivatives are computed.

Of course, in a numerical approximation the Fourier series must be truncated. For example the wall-normal component v of the velocity vector is represented as:

$$v(x,z,y,t)=\sum_{h=-\frac{N_x}{2}}^{+\frac{N_x}{2}}\sum_{l=-\frac{N_z}{2}}^{+\frac{N_z}{2}}\hat{v}_{hl}(y,t)e^{i\alpha x}e^{i\beta z} \tag{4}$$

where:

$$\alpha=\frac{2\pi h}{L_x}=\alpha_0 h;\qquad \beta=\frac{2\pi l}{L_z}=\beta_0 l$$

Here h and l are integer indexes corresponding to the streamwise and spanwise direction respectively, and α_0 and β_0 are the fundamental wavenumbers in these directions, defined in terms of the streamwise and spanwise lengths L_x and L_z of the computational domain. The computational parameters given by the streamwise and spanwise lenght of the computational domain, L_x and L_z, and the truncation of the series, N_x and N_z, must be chosen so as to minimize computational errors.

In the numerical solution of equations (2) and (3), FFT algorithms are used to compute the nonlinear terms exactly in a computationally efficient manner. Dealiasing is performed by expanding the number of collocation points by a factor 3/2 before going from the Fourier space into the physical space, to avoid the introduction of spurious energy from the high-frequency into the low-frequency modes during the calculation.

After having solved, at each time step, equations (2) and (3), the remaining two velocity components, needed for the evaluation of the nonlinear terms at later times, can be easily computed by solving for each wave number pair the 2x2 algebraic system formed by the definition (1) of $\hat{\eta}$ and by the continuity equation:

$$\begin{cases}\hat{u}_{hl}=\dfrac{1}{k^2}\left(i\alpha D(\hat{v}_{hl})-i\beta\hat{\eta}_h\right)\\[2ex]\hat{w}_{hl}=\dfrac{1}{k^2}\left(i\alpha\hat{\eta}_{hl}+i\beta D(\hat{v}hl)\right)\end{cases}$$

This system is singular when $k^2=0$, reflecting the fact that for $\alpha=0$ and $\beta=0$ is $\hat{v}_{00}(y,t)\equiv 0$ and $\hat{\eta}_{00}(y,t)\equiv 0$. When $k^2=0$ the velocity components $\hat{u}_{00}(y,t)$ and $\hat{w}_{00}(y,t)$ must be computed directly, by solving the relevant components of the momentum equation. For both directions, the computations can be performed either for a given flow rate or pressure gradient.

Time integration of the equations is performed by a partially-implicit method, as described for example in [1], so that the explicit part of the equations can benefit from a higher-accuracy scheme, while the stability-limiting implicit part is subjected to an implicit advancement. Just as in [1], an explicit third-order, low-storage Runge-Kutta method is used for the integration of the explicit part of the equations. An implicit second-order Crank-Nicholson scheme is used for the explict part.

3.1 High-accuracy finite differences schemes

The discretization of the equations in the wall-normal direction is performed through compact finite differences (FD) schemes with fourth-order accuracy over a computational molecule composed of five arbitrarily spaced grid points.

The basic idea of compact schemes can be most easily understood by thinking of a standard FD formula as a polynomial interpolation of a trascendent function in Fourier space, with the degree of the polynomial corresponding to the order of accuracy of the FD formula. Compact schemes improve the interpolation by replacing the polynomial with a ratio of two polynomials, i.e. with a rational function. This obviously increases the number of available coefficients, and moreover gives control over the behavior at infinity (in frequency space) of the interpolant, while a polynomial necessarily diverges. This allows a compact FD formula to approximate a differential operator in a wider frequency range.

Let us consider an nth-order one-dimensional ordinary differential equation in the form:

$$D_n\left(a_n f\right) + D_{n-1}\left(a_{n-1} f\right) + ... + D\left(a_1 f\right) + a_0 f = g \tag{5}$$

where the coefficients a_i are arbitrary functions of the independent variable y. Let us moreover suppose that a differential operator, for example D_n, is approximated in frequency space as the ratio of two polynomials, say $\mathcal{D}_n$ and $\mathcal{D}_0$. Polynomials like $\mathcal{D}_n$ and $\mathcal{D}_0$ have their counterpart in physical space, and we indicate with d_n and d_0 the corresponding FD operators. Let us additionally suppose that *all* the differential operators appearing in the example equation (5) admit a representation such as the preceding one, in which the polynomial $\mathcal{D}_0$ at the denominator remains the same. This basic hypothesis allows equation (5) to be recast in the discrete form:

$$d_n\left(a_n f\right) + d_{n-1}\left(a_{n-1} f\right) + ... + d_1\left(a_1 f\right) + d_0\left(a_0 f\right) = d_0\left(g\right)$$

and to be solved with the standard FD operators, provided the FD operator d_0 is applied to $a_0 f$ and to the right-hand-side g.

Now, it happens that the basic requirement of having the same $\mathcal{D}_0$ for a five-point stencil is always satisfied for a second-order equation, and is also satisfied for a fourth-order equation when the third derivative is missing, as highlighted first in the seminal work by [9]. This is the case indeed for equations (2) and (3).

The actual computation of the coefficients of the FD formulas for a certain order p of accuracy, following a standard procedure in the theory of Padè approximants, is performed by choosing a set t of polynomials of y of increasing degree, by analitically calculating their derivatives $D_n(t)$, and by imposing that the discrete equation:

$$d_n\left(t\right) = d_0\left(D_n(t)\right)$$

is exactly verified for:

$$t(y) = 1, y, y^2, ..., y^m \qquad m \geq n + p - 1$$

The solution of an m-order linear system thus gives the coefficients of the operators d_n (the highest derivative) and d_0. The coefficients of the derivatives of lesser degree are derived by analogous relations.

The use of a variable mesh size in the wall-normal direction in such a way as to still keep a fourth-order accuracy requires this procedure to be performed numerically at each y station, but only at the very beginning of the computations. Considering that $m = \mathcal{O}(10)$ and the number of y stations is $\mathcal{O}(100)$, the computer-based solution of the systems require a computing time of the order of fractions of a second, compared with a total computing time of the order of days.

3.2 Parallel computing with a cluster of PCs

Parallel computing is an almost mandatory requisite for a modern DNS code. Among the possible

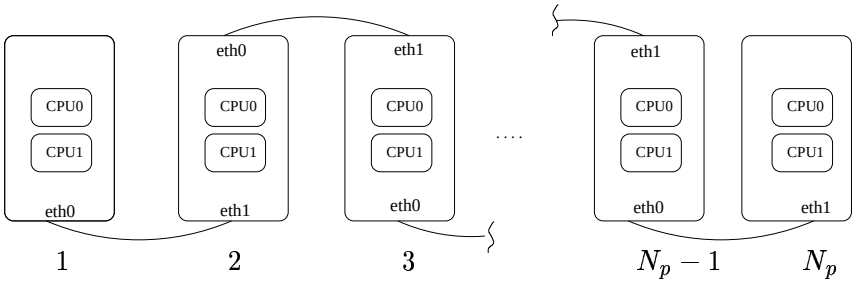

Figure 2: Connection topology of the PC cluster at DIAPM

strategies for the design of a parallel computer code, we decided to favour the availability of a large amount of CPU time at low or no cost; hence we did not consider supercomputer-type or massively parallel machines, and wrote a parallel code focused toward the use of commodity hardware, in particular a cluster of SMP Personal Computers. We currently run our DNS code on a PC cluster purposely built at the Dipartimento di Ingegneria Aerospaziale at Politecnico di Milano (DIAPM). The cluster is composed of 8 SMP PCs. Each node is equipped with 2 Pentium III 733MHz CPU and 256MB 133MHz SDRAM; the nodes are connected each other with two cheap Fast Ethernet cards.

The low-cost, 2 CPU SMP motherboards constituting the single nodes of the cluster can be exploited, obtaining high parallel efficiencies, by using the standard shared-memory facilities of a unix system, and "forking" new processes as needed. The operating system itself then handles the assignment of tasks to different CPUs, and only task syncronization is a concern at the programming level.

The use of multiple machines with distributed memory (the cluster nodes) requires communication between different machines, since the data set is spread between the computing machines. If N_p machines take part to the computation, each machine stores one of the N_p slices of the computational domain, cut by planes parallel to the walls. Thus in the computing intensive part of the evaluation of the nonlinear terms no communication is needed at all, since the FFTs are performed along x, z planes. Communication is needed when the linear systems arising from the implicit part of the equations are solved, and the recovery of the u and w velocity components is needed. Thanks to the choice of finite differences for the discretization of the y direction, the amount of communication is low, and any node of the cluster needs to exchange data only with its nearest neighbours. This allows a very simple connection topology, schematically illustrated in figure 2, where each machine is connected through two dedicated ethernet cards to the previous machine and to the next. The necessity of a hub or switch is thus eliminated, increasing simplicity and cost-effectiveness.

Most of the parallel-specific code lines reside in the general routines which solve a linear system: the burdening in the source code is reduced to a minimum. Instead of using message-passing libraries, for inter-node communication we rely directly on basic OS networking services: unix sockets and the TCP/IP protocol. The very low impact of communication times on the performance of the code (see § 6 below) is an *a posteriori* justification of this choice and of the related decision of using cheap standard hardware (100 MBits Ethernet cards) for inter-node communication.

4. CYLINDRICAL COORDINATES: THE GOVERNING EQUATIONS

The Navier–Stokes equations in cylindrical coordinates are manipulated in such a manner as to yield a two-equation formulation of the implicit part of the algorithm quite similar to its cartesian counterpart. The way this is achieved is reported in some detail in this Section.

The cylindrical coordinate system used in the present work is illustrated in figure 1. The flow between two concentric cylinders is assumed to be periodic in the axial and azimuthal directions. The inner cylinder has radius $\mathcal{R}_i$ and the outer cylinder has radius $\mathcal{R}_o$. The reference length δ is taken to be one half of the gap width. The reference velocity is chosen to be the bulk velocity U_b, so that a

Reynolds number can be defined as $\mathsf{Re} = \frac{U_b 2\delta}{\nu}$, where ν is the kinematic viscosity of the fluid.

The components of the non-dimensional Navier–Stokes equations for an incompressible fluid in cylindrical coordinates are coupled through the viscous and convective terms; therefore a fully implicit treatment of the viscous terms, as usually done in the cartesian case, will not be possible. Once the periodicity assumption is made both in the axial and azimuthal directions, the equations of motion can be conveniently Fourier-transformed along the x and θ coordinates. The symbols α and m denote the axial and azimuthal wave numbers, respectively. By defining $k^2 = (m/r)^2 + \alpha^2$, and by introducing the following notation:

$$D(f) = \frac{\partial f}{\partial r}; \qquad D_*(f) = \frac{\partial f}{\partial r} + \frac{f}{r},$$

the Laplacian operator in cylindrical coordinates can be written in the more compact form:

$$\nabla^2 = D_* D - k^2$$

The main difference between the transformed equations and the analogous equations in cartesian coordinates is the fact that k^2 depends on r. As a consequence thereof, k^2 does not commute with the operators for radial derivatives.

Following a procedure which resembles that of the cartesian case, an equation for the radial component η of the vorticity vector, which does not involve pressure, can be written by taking the radial component of the curl of the momentum equation, obtaining the following second-order equation for $\hat{\eta}$:

$$\frac{\partial \hat{\eta}}{\partial t} = \frac{1}{\mathsf{Re}} \left(D D_*(\hat{\eta}) - k^2 \hat{\eta} - \frac{\hat{\eta}}{r^2} + 2\frac{im}{r} D(\hat{u}) + 2\frac{m\alpha}{r^2}\hat{v} \right) + \frac{im}{r} H_{\hat{u}} - i\alpha H_{\hat{w}} \tag{6}$$

The two boundary conditions needed for the solution of (6) are $\hat{\eta} = 0$ at $r = \mathcal{R}_i$ and $r = \mathcal{R}_o$.

This equation has an overall structure which is analogous to that of the corresponding cartesian equation (2), except that it is not independent from $\hat{v}$. Moreover, a curvature term proportional to the first radial derivative of $\hat{u}$ appears.

The derivation of an equation for the radial component $\hat{v}$ of the velocity, again without pressure terms, is less straightforward, and requires explicit use of the continuity equation in order to obtain an expression for $\hat{p}$ as a function of the velocity components.

The first step consists in taking the time derivative of the Fourier-transformed continuity equation, and in substituting the expressions for the time derivatives of $\hat{u}$ and $\hat{w}$; continuity can be then directly invoked for substituting some terms. To carry on the procedure, one needs further to use the relations obtained by applying the operator D/r and the operator D_2 to the continuity equation; after some algebra, it is possible to obtain an expression for $\hat{p}$, which, once differentiated with respect to the radial coordinate and then substituted into the v component of momentum equation, eventually gives the following form of the fourth-order equation for $\hat{v}$:

$$\begin{aligned} \frac{\partial}{\partial t}\left[\hat{v} - D\left(\frac{1}{k^2} D_*(\hat{v})\right)\right] = \frac{1}{\mathsf{Re}} D \left\{ \frac{1}{k^2} \left[k^2 D_*(\hat{v}) - D_* D D_*(\hat{v}) - 2\frac{m^2}{r^3}\hat{v} + \right.\right. \\ \left.\left. 2\frac{im}{r^2} D(\hat{w}) - 2\frac{im}{r^3}\hat{w} \right] \right\} + \frac{1}{\mathsf{Re}} \left(-k^2 \hat{v} + D D_*(\hat{v}) - 2\frac{im}{r^2}\hat{w} \right) + \\ D\left[\frac{1}{k^2} \left(i\alpha\, H_{\hat{u}} + \frac{im}{r} H_{\hat{w}} \right) \right] + H_{\hat{v}} \end{aligned} \tag{7}$$

The four boundary conditions needed for the solution of (7) are $\hat{v} = 0$ and $\partial\hat{v}/\partial r = 0$ at $r = \mathcal{R}_i$ and $r = \mathcal{R}_o$. Equation (7) shares with its cartesian counterpart the general structure, in particular the fact that it is independent of $\hat{\eta}$. Curvature terms proportional to $\hat{w}$ and to its first radial derivative are present.

5. CYLINDRICAL COORDINATES: THE NUMERICAL METHOD

The numerical method in the cylidrical coordinate system is similar to that described in §3 for the cartesian geometry. Only the main differences will be illustrated here.

The equations for $\hat{\eta}$ and $\hat{v}$ can still be solved sequentially, provided equation (7) for $\hat{v}$ is solved first, since it is independent from the other unknow $\hat{\eta}$.

The variables are expanded in finite Fourier series. For example the radial component v of the velocity vector is represented as:

$$v(x,\theta,r,t) = \sum_{h=-\frac{N_x}{2}}^{+\frac{N_x}{2}} \sum_{l=-\frac{N_\theta}{2}}^{+\frac{N_\theta}{2}} \hat{v}_{hl}(r,t) e^{i\alpha x} e^{im\theta} \tag{8}$$

where:

$$\alpha = \frac{2\pi h}{L_x} = \alpha_0 h; \qquad m = \frac{2\pi l}{L_\theta} = m_0 l$$

Here h and l are integer indexes corresponding to the axial and azimuthal direction respectively, and α_0 and m_0 are the fundamental wavenumbers in these directions, defined in terms of the axial length L_x of the computational domain and its azimuthal extension L_θ, expressed in radians.

At each time step, the remaining two velocity components are computed by solving for each wave number pair the 2x2 algebraic system formed by the definition of $\hat{\eta}$ and by the continuity equation; the obtained system is singular when $k^2 = 0$: in this case the velocity components $\hat{u}_{00}(r,t)$ and $\hat{w}_{00}(r,t)$ must be computed directly.

The cylindrical equations are advanced in time by using the same partially implicit scheme described for the cartesian case (a second-order Crank-Nicholson scheme is used for the implicit part, and a third-order Runge-Kutta method advances the explicit part), even if the explicit part is composed by nonlinear terms and by additional viscous curvature terms. It can be envisaged that curvature terms, which contain low-order derivatives, do not compromise the stability of the time scheme. Our results support this view, and indeed we have been able to solve numerically the equations using a time step comparable with that of the planar case, without incurring in stability limitations.

5.1 High-accuracy finite difference schemes

The extension of the method described in §3, which achieves IV order accuracy over a 5 unevenly spaced points stencil, is not immediate. Equation (7) for $\hat{v}$ is still fourth-order, but the highest operator is not a simple fourth radial derivative; moreover, there are third derivatives, thus preventing the possibility of finding compact schemes sharing the same denominator $\mathcal{D}_0$ in Fourier space. Last, both equations (6) and (7) do contain r-dependent coefficients not in the innermost position.

All the r-dependent coefficients in the middle of radial derivatives can be moved at the innermost position of the radial operators, as required by the example equation (5), by operating repeated integrations by parts, i.e. repeatedly performing the following substitutions, where a indicates the generic r-dependent coefficient:

$$aD(f) = D(af) - D(a)f; \qquad aD_*(f) = D_*(af) - D(a)f$$

The third derivatives in equation (7) can be directly removed by making use of the continuity equation, which allows to substitute the first radial derivative of $\hat{v}$ with terms not containing radial derivatives. This leads to the final, rather long form of the equations for $\hat{v}$ and $\hat{\eta}$, which are written in full detail in [4]. It is important to note that this introduces many additional coefficients, which are function of the radial coordinate and of the wavenumbers. With some overhead in CPU time they can be computed on the fly during the execution of the program; alternatively, they can be precomputed once at the beginning at the expense of some memory space, if the available storage allows.

The actual calculation of the FD operators of fourth order accuracy on a five point stencil can still be performed in the relatively straightforward way described in §3, provided one observes that the only way fourth derivatives enter the equations is through the operator DD_*DD_*. Thus, given a set of polynomials $t(r)$ of increasing degree in the variable r, the corresponding set of derivatives $DD_*DD_*(t)$ can be computed analitically. A linear system furnishing the coefficients of the FD operators dd_*dd_* and d_0 follows from the condition that:

$$dd_*dd_*(t) - d_0(DD_*DD_*(t)) = 0$$

5.2 The spatial resolution in the azimuthal direction

In the cylindrical coordinate system, the azimuthal extension of the computational domain decreases with r; if the necessary spatial resolution (for example the number of Fourier modes, or the collocation points in a finite-difference calculation) is set up based on the most demanding region of the flow field, i.e. the outer wall, then the spatial resolution becomes unnecessarily high when the inner wall is approached. This not only implies a waste of computational resources, but might pose also numerical stability problems.

We have adopted a truncation of the Fourier series in the azimuthal direction which is function of the radial position. Instead of the classical expansion (8), we use the following representation:

$$v(x,\theta,r,t) = \sum_{h=-\frac{N_x}{2}}^{+\frac{N_x}{2}} \sum_{l=-\frac{N_\theta(r)}{2}}^{+\frac{N_\theta(r)}{2}} \hat{v}_{hl}(r,t) e^{i\alpha x} e^{im\theta}$$

where the number of Fourier modes in the azimuthal direction is a generic function of the radial coordinate.

For the problem under consideration, the function $N_\theta(r)$ has been chosen to be linear with r, from a maximum value $N_{\theta,max}$ which guarantees an adequate spatial resolution at $r = \mathcal{R}_o$ down to a minimum $N_{\theta,min}$ which gives *the same* spatial resolution at $r = \mathcal{R}_i$. This is equivalent to assume that the Fourier modes $\hat{v}_{hl}$ with $|l| \leq N_{\theta,min}$ are defined through the whole annular gap, i.e. for $\mathcal{R}_i \leq r \leq \mathcal{R}_o$, while any mode $\hat{v}_{hl}$ with $N_{\theta,min} < |l| \leq N_{\theta,max}$ starts at $r = \mathcal{R}_o$ but ends being non null at a radial position $\overline{r}(l)$, function of the index l, intermediate between the two walls. Of course modes $\hat{v}_{hl}$ with $|l| > N_{\theta,max}$ are zero everywhere in the channel, owing to the finite representation of the variable. There is no need to introduce a special treatment of the modes with $N_{\theta,min} < |l| \leq N_{\theta,max}$ as far as the boundary conditions at $r = \mathcal{R}_i$ are concerned: these modes are simply assumed to be zero from the position $\overline{r}$ down to $\mathcal{R}_i$.

From the point of view of computer programming, a representation of the coefficients of Fourier series whose truncation varies with r has been implemented with a suitable memory management, where a two-dimensional array of pointers is used to reference into a variable-sized one-dimensional array, storing all the nonzero coefficients as a function of r, from $r = \mathcal{R}_o$ down to $\overline{r}$.

This procedure is able to reduce the computational cost of DNS in cylindrical geometries, thanks to the reduction in the number of active Fourier modes, and to avoid the stability problems deriving from the clustering of grid points in the near-axis region.

6. RESULTS

In this Section we present some results concerning the performances of the computer code based on the numerical methods discussed above. Performances can be assessed in terms of memory requirements, CPU time and parallel speedup, by comparison with a similar cartesian DNS code. We chose for the comparison the code described in [8], a recent work conducted by one of the world leading groups working on DNS, and focused on the optimization of a very similar code for parallel computing.

The RAM requirements of our (natively double-precision) code is essentially dictated by the size of the 3-dimensional arrays, can be estimated by $5N_xN_yN_z$ words. This compares with the code

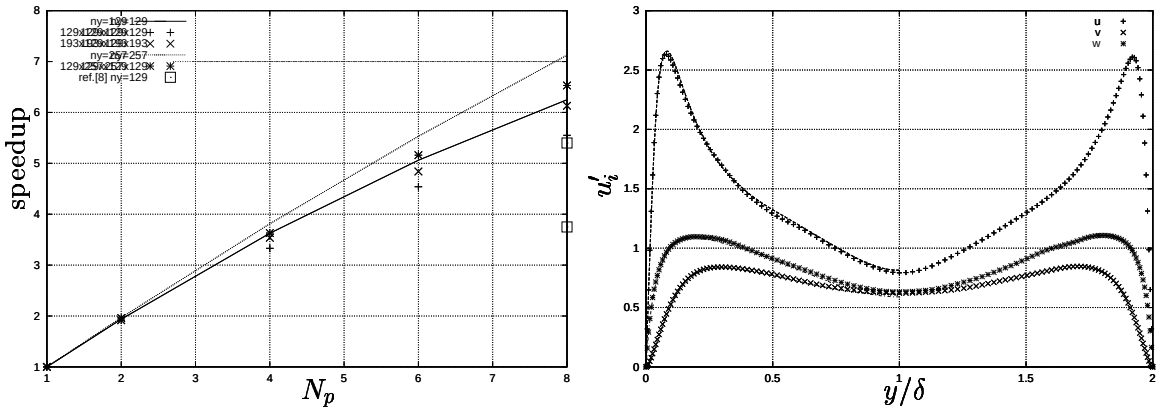

Figure 3: Left: speedup as a function of the number N_P of nodes. Lines without symbols refer to the ideal speedup. Right: root-mean-square values of velocity fluctuations across the plane channel. Lines are reference data from [5].

described in [8], explicitely optimized for memory occupation, which in minimal-memory configuration requires approximately $7N_xN_yN_z$ words, i.e. over 40% higher. The code used by [1], where memory requirements were critical, reportedly has a size similar to that of [8]. The test case sized $129x97x129$ requires in our code 74 MBytes of RAM.

This test case requires 34 CPU seconds for the computation of a full temporal step on a single processor of the cluster. This is achieved without any particular optimization or tuning of the code towards the particular architecture. As a comparison, one can deduce from [8] that the same computational case can be run, after extensive optimization, on a single processor of the 256 nodes Cray T3E of the National Supercomputer Center of Linkoeping (Sweden) with 30 CPU seconds, and on a single processor of the 152 nodes IBM SP2 machine, available at the Center for Parallel Computers of KTH University, with 7.5 seconds of CPU time. This single-node comparison shows that the present computer code runs on a commodity PC with performance fully comparable with those of expensive supercomputers.

We have to date evaluated the SMP parallel efficiency of our code with 2 CPU machines only. With processors and memory chips on the market approximately one year ago (fall 2000), the speedup was nearly linear compared to the single-processor case: this is the case, for example, with a Pentium II 500MHz PC with 2 CPUs and 512MB of 100MHz DRAM. With more recent hardware, the SMP performance is somewhat degraded to a 155% speedup, the main reason being the faster CPU clock, which makes the memory contention problem more important. This is a noticeable gain, since it is at essentially no cost.

The parallel performances of the code are illustrated in Figure 3 (left), where speedup ratios are reported as a function of the number N_p of computing nodes. Speedup is defined as the ratio between the actual computing time and the computing time needed for a single-node calculation. The continuous lines refer to the maximum ideal performances attainable for a given number of points N_y in the y direction. This ideal performance is slightly less that a linear speedup; this is due to the duplication of four planes at the interface between slices of data residing in different nodes. This reflects the focus towards a cluster made of a limited number of PCs, and not a massively parallel system, the latter necessarily implying either massive investments in hardware or nonproprietary hardware. The maximum possible speedup increases with the ratio N_y/N_p, and is well estimated by the formula:

$$speedup = N_p \left(1 - \frac{4(N_p - 1)}{N_y}\right)$$

Compared to this maximum, performances are extremely good, and become even better when the

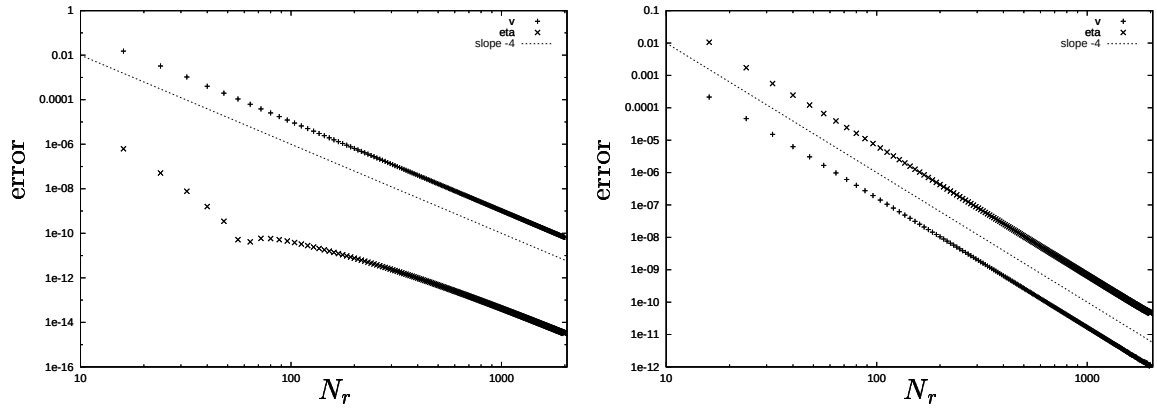

Figure 4: Difference between the FD approximation of the implicit (left) and explicit (right) parts of equations (6) and (7), applied to an arbitrary set of test functions, and the analytical solutions. Dotted lines indicate a -4 slope.

size of the problem increases. The communication time is not a bottleneck, despite the enforcing of communication with the overhead of the TCP protocol. With $N_p = 8$, the communication time is as low as 7% of the total computing time for the most demanding case of $N_x = 129$, $N_y = 257$ and $N_z = 129$, and is 14% for the worst case of $N_x = 129$,$N_y = 129$ and $N_z = 129$, which requires 6.5 seconds for iteration with a speedup of 5.5. For comparison, consider that the same test case runs in approximately 2 seconds at iteration when 8 processors are used on the $SP2$ (speedup 3.75), while it requires 7.5 seconds when 8 processors of the Cray T3E are used (speedup 5.5).

As an example, we show in figure 3 (right) some typical result of a DNS computation of the turbulent plane channel flow, in particular the distribution, in the direction normal to the wall, of the root-mean-square value of the velocity fluctuations. These results are from a benchmark case where we tryed to duplicate the results contained in [5], where the calculations by [1] have been repeated. The results are in perfect agreement. The essential observation is that we have performed such calculations overnight with our cluster.

The cylindrical version of the computer code shares with its cartesian counterpart its basic structure, including the high computational efficiency when executed in serial, SMP and parallel mode. The differences in source code are very limited, allowing to reuse most of the numerical routines. For a problem of the same computational size, the overhead of the cylidrical version is approximately 40%. Precomputing the r-dependent coefficients increases memory requirements by 13%.

An accuracy test of the method is reported, by applying the discrete operators constituted by the implicit and explicit parts of equations (6) and (7) to an arbitrarily chosen right-hand-side. The same calculation has been then repeated analitycally, and the two solutions compared. In figure 4 the difference between the analytical and discrete results are plotted as a function of the number N_r of points in radial direction. As expected, the error decreases with a slope never less than -4 in log scale; hence it can be concluded that the numerical method is actually fourth-order accurate.

In figure 5 some turbulence statistics are reported concerning the turbulent flow in an annular pipe at $\mathsf{Re} = 5600$, computed with a preliminary version of the program where the fourth-order derivative was only second-order accurate and only a single SMP machine was used. The turbulent flow in an annular pipe has never been computed with DNS hiterto, and it was believed that turbulence statistics differ from those of the plane channel flow only for high values of transversal curvature, i.e. for $\delta/\mathcal{R}_i \gg 1$, where δ is half the annular gap and $\mathcal{R}_i$ the radius of the inner cylinder. It has been found that this is not the case. The calculations reported in the figures are for $\delta/\mathcal{R}_i = 1$, and the effects of transverse curvature can be appreciated: the mean velocity profile presents a logarithmic region of lesser extent and slope; the rms velocity fluctuations are modified in the inner part of the pipe also in the near-wall region. These calculations have used more than 16 millions of degrees

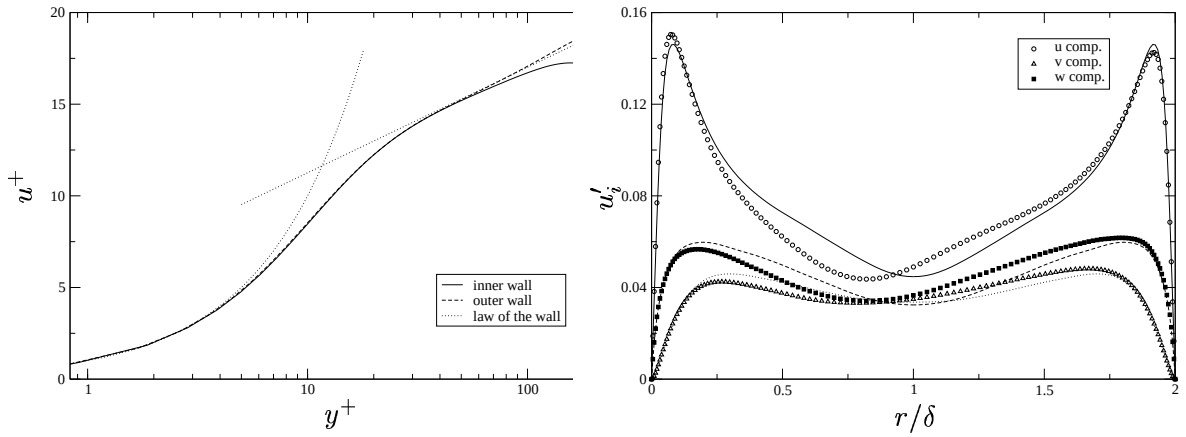

Figure 5: Mean velocity profile in wall units (left) and root-mean-square values of velocity fluctuations (right) across the annular pipe.

of freedom, with RAM requirements of 410MB; the computing time was about 4 minutes for each complete timestep.

ACKNOWLEDGMENTS

Financial support from ASI in 1999, and from MURST in 1999 and 2000 is acknowledged. The Authors whish to thank ing. Patrick Morandi for his help in part of the work concerning cylindrical coordinates.

References

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

[2] B. Ma, Z. Zhang, F. Nieuwstadt, and C. van Doorne. On the spatial evolution of a wall-imposed periodic disturbance in pipe Poiseuille flow at Re=3000. Part 1. Subcritical disturbance. *Journal of Fluid Mechanics*, 398:181–224, 1999.

[3] P. Moin and K. Mahesh. Direct numerical simulation: A tool in turbulence research. *Annual Review of Fluid Mechanics*, 30:539–578, 1998.

[4] P. Morandi and M. Quadrio. On the numerical solution of the Navier–Stokes equations in cylindrical coordinates. Technical Report DIA-SR 01-01, Dip. Ing. Aerospaziale Politenico di Milano, 2001.

[5] R. Moser, J. Kim, and N. Mansour. Direct numerical simulation of turbulent channel flow up to $Re_\theta = 590$. *Physics of Fluids*, 11(4):943–945, 1999.

[6] J. Neves, P. Moin, and R. Moser. Effects of convex transverse curvatuire on wall-bounded turbulence. Part 1. The velocity and vorticity. *Journal of Fluid Mechanics*, 272:349–381, 1994.

[7] P. Orlandi and M. Fatica. Direct simulations of turbulent flow in a pipe rotating around its axis. *Journal of Fluid Mechanics*, 343:43–72, 1997.

[8] A. Lundbladh, S. Berlin, M. Skote, C. Hildings, J. Choi, J. Kim and D.S. Henningson. *An efficient method for simulation of incompressible flow over a flat plate.* TRITA-MEK 1999:11, 1999.

[9] L. Thomas. The stability of plane Poiseuille flow. *Physical Review*, 91(4):780–783, 1953.