Computer Physics Communications 141 (2001) 365–374

Computer Physics Communications

www.elsevier.com/locate/cpc

A new technique for a parallel dealiased pseudospectral Navier–Stokes code

Michele Iovieno [a], Carlo Cavazzoni [b], Daniela Tordella [a,*]

[a] *Dipartimento di Ingegneria Aeronautica e Spaziale, Politecnico di Torino, C.so Duca degli Abruzzi 24, 10129 Torino, Italy*
[b] *CINECA, Consorzio Interuniversitario, Via Magnanelli 6/3, 40033 Casalecchio di Reno (Bo), Italy*

Received 15 August 2001; accepted 22 September 2001

Abstract

A novel aspect of a parallel procedure for the numerical simulation of the solution of the Navier–Stokes equations through the Fourier–Galerkin pseudospectral method is presented. It consists of a dealiased ("3/2" rule) transposition of the data that organizes the computations in the distributed direction in such a way that whenever a Fast Fourier Transform must be calculated, the algorithm will employ data stored solely on the proper memory of the processor which is computing it. This provide for the employment of standard routines for the computations of the Fourier transform. The aliasing removal procedure has been directly inserted into the transposition algorithm. The code is written for distributed memory computers, but not specifically for a peculiar architecture. The use on a variety of machines is allowed by the adoption of the Message Passing Interface library. The portability of the code is demonstrated by the similar performances, in particular the high efficiency, that all the machines tested show up to a number of parallel processors equal to $1/2$ the truncation parameter $N/2$. Explicit time integration is used. The present code organization is relevant to physical and mathematical problems which require a three dimensional spectral treatment. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Navier–Stokes equations; Spectral methods; Parallel; Transposition algorithm

1. Introduction

Numerical flow field simulations based on spectral methods have been often carried out in the last decades. A detailed review on these methods is given by Canuto et al. [5] and by Fisher and Patera [10]. Full three-dimensional pseudospectral codes are conveniently used in association with periodic type of boundary conditions for the direct simulation of unsteady free flows such as homogeneous isotropic turbulence (Gagne and Castaing, 1991 [11], Jackson et al., 1991 [14], Jimenez et al., 1993 [15]), free mixing layers (Briggs et al., 1996 [3])—which know applications in astrophysics (Ray, 1996 [22])—and homogeneous turbulence subject to solid body rotation (Cambon et al., 1997 [4]), which is of concern in geophysics.

Even if spectral methods need less mesh points that other spatial discretization methods to achieve a given accuracy, parallel processing in such a situations may be required due to the hugeness of computational resources entailed. The global character of the multidimensional Fourier transforms, repeatedly computed within any spectral algorithm, constitutes the

[*] Corresponding author.
E-mail addresses: iovieno@athena.polito.it (M. Iovieno), c.cavazzoni@cineca.it (C. Cavazzoni), tordella@polito.it (D. Tordella).

main difficulty inside the process of parallelization of a Navier–Stokes code.

Examples of flows simulations exploiting a large parallelism, mostly performed in the last fifteen years, are described in the review paper by Fisher and Patera [10]. Here we will mention briefly few applications that implemented a spectral method in all the directions. In 1985 Dang [9] evaluated, versus direct numerical computations done over 64^3 points, simple sub-grid-scale turbulent models for the simulation of homogeneous and strained turbulence over 16^3 points, using two array processors. In 1988 Canuto and Giberti [6] presented a spectral algorithm for the direct simulation of homogeneous, fully developed turbulence. The algorithm is detailed with the comparative analysis of all the various factors—from differentiation in transform space to time advancing—affecting the parallel performance and with the global speed-up as a function of the local speed-up in executing the structure of wave number k over n_p processors. Homogeneous turbulence implementation on a hypercube (32 nodes iPSC/860) has been described in Jackson et al. (1991, [12]) by means of a pseudospectral non-dealiased approach in which the Fast Fourier Transforms (FFT in the following) are computed through a procedure performing cross-processor transposes along the spatial/wavenumber dimension of memory distribution. In 1991 Pelz [21] simulated homogeneous isotropic turbulence on a 1024-node hypercube decomposing the domain in all the direction and using parallel multidimensional FFT algorithm. Also in 1992 appeared one of the up to now better resolved (512^3) homogeneous isotropic turbulence simulation (Chen and Shan, 1992 [7]). The computation was done on 65536 processors of the Connection Machine-2 by means of a fully parallel algorithm. An algorithm conceived to run on three partitioned parallel machines, where to each spatial projection of the Navier–Stokes equations corresponds a group of processors, has been proposed by Basu in 1994 [1], using the 2/3 rule of dealiazing. Two parallel implementations of 3D Fast Fourier Transforms suitable for pseudo-spectral simulation of turbulent field, however specific for the IBM SP1 and SP2 scalable parallel machines, have been presented by Briscolini in 1995 [2].

In this paper, we describe a parallel algorithm for the dealiased pseudospectral Fourier–Galerkin method applied to the Navier–Stokes equations. The aliasing error is removed through the "3/2-rule", which enables the removal of the error without reducing the number of active modes. The procedure is—for the first time—combined with the computation of direct and inverse FFT through a proper transposition technique. The use of real-to-real one processor one-dimensional standard FFT routines and of MPI library (see [14,19]) allows a very good portability of the code, demonstrated through comparative benchmark results produced by three different machines.

The physical problem and the numerical method are reviewed in Section 2. In Section 3 the parallelization technique is detailed. Section 4 contains the code performances over different machines and resolutions and, as a general example of performance, the computation of the three-dimensional energy spectrum for a decaying homogeneous isotropic turbulence field. The simulation Reynolds number, based on the Taylor microscale, is that for which there are available comparisons with both experimental [8] and direct numerical simulations [15] results.

2. The physical problem

We consider flow fields represented by the incompressible Navier–Stokes equations

$$\partial_i u_i = 0, \tag{1}$$

$$\partial_t u_i + \partial_j (u_i u_j) = -\rho^{-1} \partial_i p + \nu \nabla^2 u_i, \tag{2}$$

where $u_i(x_1, x_2, x_3, t)$ is the velocity field, $p(x_1, x_2, x_3, t)$ the pressure, ρ the constant density and ν the kinematic viscosity, that is also assumed constant. The domain is cubic of size L. The boundary conditions are periodic and represented by the relationships:

$$u_i\big(\mathbf{x} + L\mathbf{e}^{(j)}, t\big) = u_i(\mathbf{x}, t) \quad \forall i, j. \tag{3}$$

These boundary conditions are particularly suited to simulate homogeneous and isotropic, mono or multiphase, turbulent fields either by means of the direct simulation technique or by means of the large eddy scale method (see, for example, Lesieur and Metais [16], Meneveau and Katz [18]), but they may also be used for other kind of free flows and are not limited to turbulent regimes.

The mathematical structure of the problem leads naturally to the adoption of a spectral discretization, and in particular to a Fourier–Galerkin method, a weighted residuals method in which the basis and test functions are both trigonometric polynomials of degree $\leqslant N/2$. The spectral method is the most accurate method since using it to compute an infinitely smooth solution the numerical error decays exponentially rather than algebraically as the resolution is increased [5].

Taking the divergence of (2) and substituting (1), the problem may be conveniently rewritten as

$$\partial_t u_i = -\partial_j(u_i u_j) - \rho^{-1}\partial_i p + \nu\nabla^2 u_i, \tag{4}$$

$$-\rho^{-1}\nabla^2 p = \partial_i\partial_j(u_i u_j). \tag{5}$$

The expansion of u_i and p in terms of the basis functions are then introduced (see [5]),

$$u_i^N(\mathbf{x},t) = \sum_{k_1,k_2,k_3=-N/2}^{N/2-1} \hat{u}_{i,\mathbf{k}}^N(t)\mathrm{e}^{\mathrm{i}\mathbf{k}\cdot\mathbf{x}}, \tag{6}$$

$$p^N(\mathbf{x},t) = \sum_{k_1,k_2,k_3=-N/2}^{N/2-1} \hat{p}_{\mathbf{k}}^N(t)\mathrm{e}^{\mathrm{i}\mathbf{k}\cdot\mathbf{x}} \tag{7}$$

and the solution is enforced by requiring that the residual of Eqs. (4), (5) be orthogonal to all the test functions. This yields:

$$\partial_t \hat{u}_{i,\mathbf{k}}^N = -\mathrm{i}k_j\widehat{(u_j u_i)}_{\mathbf{k}}^N - \mathrm{i}k_i\rho^{-1}\hat{p}_{\mathbf{k}}^N - \nu k^2\hat{u}_{i,\mathbf{k}}^N, \tag{8}$$

$$k^2\rho^{-1}\hat{p}_{\mathbf{k}}^N = -k_i k_j\widehat{(u_i u_j)}_{\mathbf{k}}^N, \quad k = \|\mathbf{k}\|. \tag{9}$$

The pressure may be eliminated to yield

$$\partial_t \hat{u}_{i,\mathbf{k}}^N = -\mathrm{i}k_j\widehat{(u_j u_i)}_{\mathbf{k}}^N + \mathrm{i}k_i\frac{k_l k_j}{k^2}\widehat{(u_l u_j)}_{\mathbf{k}}^N - \nu k^2\hat{u}_{i,\mathbf{k}}^N. \tag{10}$$

The semi-discrete equation (10) is then resolved using a four-stages fourth-order explicit Runge–Kutta scheme in the low storage version by Jameson, Schmidt and Turkel (1981) [13].

3. Code parallelization

3.1. Parallelization strategy

The code uses as main unknowns the Fourier coefficients of the expansion (6), which are stored in Hermitian form in each direction. As opposed to other spatial discretization schemes, like finite difference or finite volume, most operations are local in the wave numbers space and do not require the knowledge of the other dependent variables in the surrounding of a given point in the physical domain. This is true both for the differentiation, that reduces to multiplication of the Fourier coefficient by the relevant wave number, and for the solution of the Poisson equation (5), that becomes now a diagonal linear system of N^3 independent equations, see Eq. (9). The dependent variable fields are stored in three-dimensional matrices that have been distributed among processors along only one direction. The algorithm does not depend on the direction and one is free to choose any of the three spatial directions to distribute the data among the processors (from now on, we conventionally indicate the direction of distribution as direction 3). This strategy is convenient because, whether associated to an opportune data transposition technique among the processors, allows computation of the right hand side of the system (9) and thus to associate to each processor a subset of relevant equations without overlapping of data. For this reason, assuming the use of processors having sufficient memory, there is no advantage in the distribution of all directions. Furthermore, leaving to each processor the full data in the two non-distributed directions increases the numerical efficiency of the FFT (unidimensional, real to real) computations.

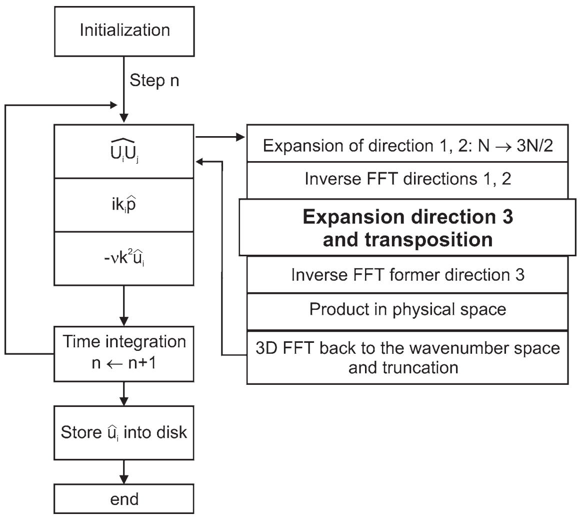
Initialization
Step n
$\widehat{U_iU_j}$
$ik_i\hat{p}$
$-\nu k^2\hat{u}_i$
Time integration n ← n+1
Store $\hat{u}_i$ into disk
end
Expansion of direction 1, 2: N → 3N/2
Inverse FFT directions 1, 2
Expansion direction 3 and transposition
Inverse FFT former direction 3
Product in physical space
3D FFT back to the wavenumber space and truncation

Fig. 1. Flow chart: n temporal index, N dimension of the data matrices.

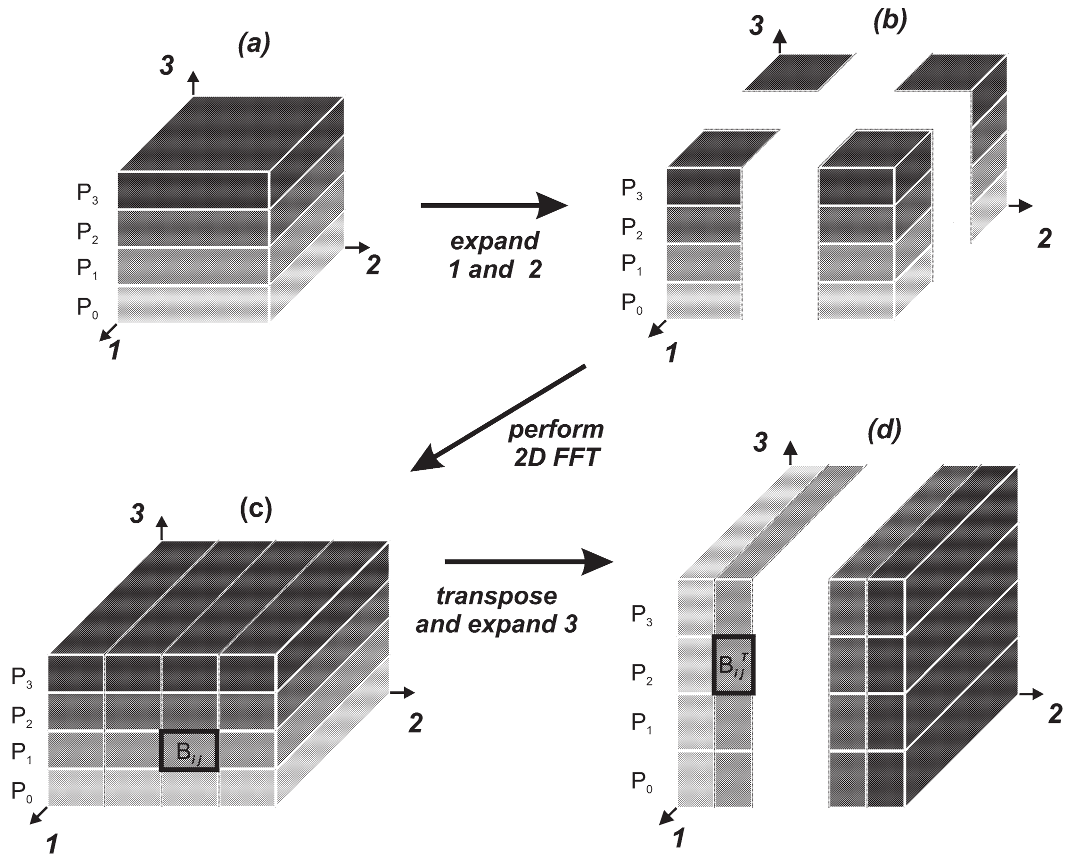

Fig. 2. Data distribution among the processors in the procedure for the computation of the dealiased product. (a) Original data in the wavenumber space, (b) data after expansion of the non distributed directions, (c) scheme after executing the 1D inverse FFT in the non distributed directions, (d) data after transposition and expansion in the former distributed direction (see §3.3). The figure exemplifies the case of 4 parallel processors: the greatest possible value of n_p is equal to the value of the truncation parameter $N/2$.

Consequently, in the code only one direction in the wavenumber space is distributed among the processors, leaving to each processor the full data in the other two directions (see Fig. 2(a)).

Let us come now more specifically to the main problem of the computation of the nonlinear term in Eq. (8) or (10). It is evaluated using the dealiased pseudospectral method: the data are first transformed in the physical space where the pointwise multiplication is performed, then the result is transformed again in the wavenumber space.

As always, the operation of transformation, performed using the FFT algorithm, is a global procedure that uses information from all the data and for this reason it is the very central point of the parallel application. In the following two paragraphs we detail the dealiased non linear term computation as well as the parallel data transposition.

We point out that the present organization of the code limits the communication events among the processors into the dealiased transposition within the FFT computation. A more classical organization based on a preliminary expansion of the data matrices in all the directions should have inevitably introduced, in the parallel context, further communication occurrences.

3.2. Dealiased pseudospectral product computation

The computation of the Fourier coefficient of the non-linear convective terms in (8) through the straightforward application of the pseudospectral transform method introduces aliasing errors that should be removed. Because the aliasing error is asymptotically no worse than the truncation error (see [5], pp. 118, 123) this procedure might be strictly necessary only when a high number of grid points cannot be used.

The computation of an aliasing error-free product may be performed using various techniques, mainly the truncation or padding technique [20] or the phase shifts [23,24] technique. The first, referred to as the $3/2$ rule, evaluates the discrete transforms over $3/2N$ points and extends to three dimensions in a straightforward manner. It is, among the procedures which remove completely the aliasing error without reducing the number of active modes, the one that in practice ($N > 16$) requires the smallest number of operations per mode, see again [5], p. 206. It consists of an expansion of the matrices containing the Fourier coefficients from a dimension equal to N to a dimension equal to $M = 3N/2$. The expansion is done adding zeros to the coefficients representing the extra wave numbers.

The product is pointwise computed in the finer mesh obtained after the computation of the inverse transform of the velocity components under multiplication. The result is transformed again in the wavenumber space using the three-dimensional FFT procedure. It is convenient to combine the expansion procedure and the inverse transform algorithms with associated transposition, because this leads to a reduced number of one-dimensional FFT computations and to a single event of communication among processors.

The procedure used is constituted by the following steps:

(1) Expansion in the non distributed directions (k_1 and k_2):

$$\tilde{u}(k_1,k_2,k_3) = \begin{cases} \hat{u}(k_1,k_2,k_3) & \text{if } |k_1,k_2| \leqslant \frac{N}{2}, \\ 0 & \text{otherwise;} \end{cases}$$

$$\tilde{v}(k_1,k_2,k_3) = \begin{cases} \hat{v}(k_1,k_2,k_3) & \text{if } |k_1,k_2| \leqslant \frac{N}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

This brings matrices of dimension $N \times N \times N/n_p$ to matrices with dimensions $M \times M \times N/n_p$, where n_p is the number of processors.

(2) Execution of the one-dimensional inverse FFT in the expanded directions (only on the 1D sub-array is which elements are not all equal to zero):

$$\tilde{u}(j_1,k_2,k_3) \leftarrow \frac{1}{M}\sum_{k_1=0}^{M-1} \tilde{u}(k_1,k_2,k_3)\mathrm{e}^{-\frac{2\pi \mathrm{i}}{N}k_1 j_1},$$

$$\tilde{u}(j_1,j_2,k_3) \leftarrow \frac{1}{M}\sum_{k_2=0}^{M-1} \tilde{u}(j_1,k_2,k_3)\mathrm{e}^{-\frac{2\pi \mathrm{i}}{N}k_2 j_2}.$$

(3) Parallel transposition and expansion in the former distributed third direction. This yields to the expanded field:

$$\tilde{\tilde{u}}(j_1,k_3,j_2) = \begin{cases} \tilde{u}(j_1,j_2,k_3) & \text{if } |k_3| \leqslant \frac{N}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

Here and in the following the double tilde symbol represents an expanded variable.

(4) Execution of the FFT in the second (former third) direction.

(5) Computation of the pointwise product

$$\tilde{\tilde{u}}(j_1,j_3,j_2) \leftarrow \tilde{\tilde{u}}(j_1,j_3,j_2)\tilde{\tilde{v}}(j_1,j_3,j_2).$$

(6) Computation of the three-dimensional FFT, leading to $\tilde{\tilde{u}}(k_1,k_2,k_3)$. This is done through the following four-stage procedure, that includes again a transposition at the third stage:

$$\tilde{\tilde{u}}(k_1,j_3,j_2) \leftarrow \sum_{j_1=0}^{M-1} \tilde{\tilde{u}}(j_1,j_3,j_2)\mathrm{e}^{\frac{2\pi \mathrm{i}}{M}k_1 j_1},$$

$$\tilde{\tilde{u}}(k_1,k_3,j_2) \leftarrow \sum_{j_2=0}^{M-1} \tilde{\tilde{u}}(k_1,j_3,j_2)\mathrm{e}^{\frac{2\pi \mathrm{i}}{M}k_2 j_2},$$

$$\tilde{\tilde{u}}(k_1,j_2,k_3) \leftarrow \tilde{\tilde{u}}(k_1,k_3,j_2),$$

$$\tilde{\tilde{u}}(k_1,k_3,k_2) \leftarrow \sum_{j_3=0}^{M-1} \tilde{\tilde{u}}(k_1,j_3,k_2)\mathrm{e}^{\frac{2\pi \mathrm{i}}{M}k_3 j_3}.$$

(7) Truncation of the data from M modes to the original N modes.

3.3. Parallel transposition

Parallel transposition occurs two times in the pseudospectral computation of the convolution sums: throughout the inverse FFT of expanded data and throughout the FFT of the result of the product computation. The first time it is associated with the data expansion, i.e. the data are transposed while being stored in an expanded 3D matrix—according to the $3/2$ rule of dealiazing—while the second time the dimension of the matrix of data storage is left unchanged. In the following, we refer primary to the first one, the scheme of the latter may be recovered leaving equal dimensions in all directions.

The transposition algorithm is implemented in a dedicated subroutine. Variable $f(k_1,k_2,k_3)$ will contain in input the data to be transposed, in output the result of the transposition.

With reference to Fig. 2, the data of each processor i are composed into elementary blocks $B_{ij}(k_1,k_2,k_3)$ that are to be transposed and transferred to the other processor. The index j indicates the logical distance among the processors. Each processor i must first identify the data block to be transferred to the processor $i+j$, then it must transpose it and the data block to be received from the processor $i-j$. A call to a routine send-receive-replace of the Message Passing Interface Library occurs and then the two transfers take place. Note that this straightforward communication pattern is functional to the portability of the code. In case this last is not required, things could be improved taking into account the underlying network topology or making use of machine specific communication libraries such as "Cray shmem".

Finally, each processor allocates the data received in the correct position, see the scheme in the following part of this paragraph.

N and $M = 3N/2$ are the original dimensions of data matrices, $n_{\text{loc}} = N/n_p$ and $m_{\text{loc}} = M/n_p$ are the correspondent dimensions—in the distributed direction—of the matrices contained in each processor. Procedure details are the following.

There is a single preliminary cycle ($j=0$) wherein each processor must transpose and collocate the data of the block B_{i0}:

$$g\big(j_1, j_2 + I(i), j_3\big) \leftarrow f(j_1, j_3 + \mathrm{i}m_{\text{loc}}, j_2)$$
$$\forall j_1, j_2 = 0..M-1,\ \forall j_3 = 0..n_{\text{loc}}-1;$$
$$I(i) = \begin{cases} n_p i & \text{if } i < n_p/2, \\ M - (n_p - i)n_{\text{loc}} & \text{otherwise.} \end{cases}$$

Then, a main loop is executed:

for $j = 1,\ n_p - 1$:

(1) Each processor defines the destination and source processors: [1]

$$i_d \leftarrow i + j, \quad i_s \leftarrow i - j.$$

(2) Each processor creates and transposes the block to be sent

$$B^{\mathrm{T}}_{ij}(j_1, j_2, j_3) \leftarrow f(j_1, j_3 + i_d n_{\text{loc}}, j_2)$$
$$\forall j_1, j_2 = 0, M-1,\ \forall j_3 = 0..n_{\text{loc}}-1.$$

(3) A send–receive operation occurs through a call to the routine MPI_send_recv_replace [19]. Each processor sends to the destination processor its block B_{ij} and receive from source processor the relevant block, which is stored in the same memory allocations that was containing the data sent.

(4) Each processor allocate the received block in the new position:

$$g\big(j_1, j_2 + I(i_s), j_3\big) = B^{\mathrm{T}}_{i_s j}(j_1, j_2, j_3)$$
$$\forall j_1, j_2 = 0, .., M-1,\ j_3 = 0..n_{\text{loc}}-1.$$

End of the loop.

4. Timings and efficiencies of the computation procedure. Validation over a physical application

Looking at the benchmark results of the code, applied to the simulation of a decaying homogeneous turbulent field, see Fig. 3, the Cray T3E appear to be the slowest of the three tested architectures, while the SGI ORIGIN and the IBM SP3 have similar performance. On the other hand, if we look at the speed up S—defined as the ratio of the run time required by one processor referred to the run time required by n_p processors—see Fig. 4, the present code on the Cray T3E displays a speed up close to the ideal one at all grid sizes. The corresponding efficiency $E = S/n_p$ is always close to 1. IBM SP3 also displays a good speed up but the curves move away from the ideal behaviour well before the curves relevant to the CRAY T3E happen to do. The efficiency for this machine results inside the interval 0.8–1.1. The speed up relative to the SGI ORIGIN shows both a strong super-linear behaviour followed

[1] A circular permutation of indexes is assumed.

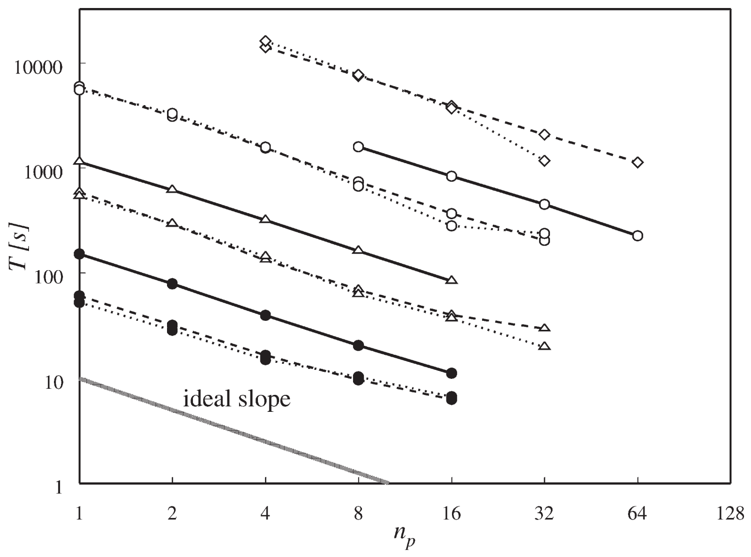

Fig. 3. Execution time as a function of n_p for ten iterations of the main code loop: (—–) CRAY T3E 1200, (·····) SGI ORIGIN 3800 R14000 500 MHz and (– – –) IBM SP3 Pwr3 nightawk II 375 MHz. Symbols refer to the different grid sizes: 32^3 (•), 64^3 (△), 128^3 (○) and 256^3 (◇).

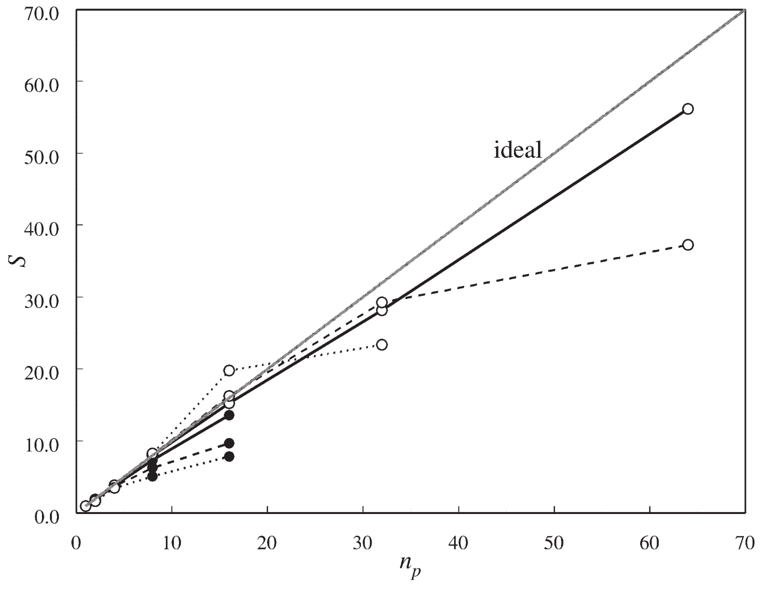

Fig. 4. Comparison of the code speed up on different architecture: (—–) CRAY T3E 1200, (·····) SGI ORIGIN 3800 R14000 500 MHz, (– – –) IBM SP3 Pwr3 nightawk II 375 MHz. Symbols refer to the grid sizes: 32^3 (•), 128^3 (○).

by a fast saturation for the 64^3 and 128^3 grids. The efficiency is in the range 0.75–1.2.

We observe that the ratio R_1 between the time spent to compute the non linear terms in (8), (9) (see in §3.2 the products of the velocity components in the space of wave numbers) and the total computation time of one time step turns out to be in the range $0.75 \sim 0.94$, independently from the machine used and the number

Table 1
First lines in each block: ratio R_1 between the time spent to compute the non linear terms and the total computation time of one time step, for different machines and number of parallel processors. Second lines in each block: ratio R_2 between the time required by the transposition procedure and the time spent to compute the non linear term

n_p	Cray T3E			SP3				ORIGIN			
	32	64	128	32	64	128	256	32	64	128	256
1	0.914	0.910		0.944	0.906	0.877		0.943	0.919	0.822	
	0.042	0.042		0.033	0.370	0.422		0.027	0.068	0.069	
2	0.909	0.912		0.943	0.910	0.878		0.940	0.909	0.788	
	0.071	0.067		0.060	0.134	0.131		0.080	0.154	0.187	
4	0.898	0.903		0.944	0.918	0.879	0.880	0.941	0.909	0.753	0.791
	0.087	0.083		0.080	0.126	0.140	0.143	0.099	0.177	0.205	0.233
8	0.885	0.893	0.886	0.929	0.919	0.890	0.879	0.917	0.912	0.876	0.758
	0.105	0.092	0.066	0.141	0.373	0.130	0.163	0.319	0.143	0.237	0.238
16	0.870	0.865	0.862	0.899	0.904	0.897	0.858	0.818	0.906	0.897	0.757
	0.134	0.099	0.072	0.265	0.218	0.152	0.145	0.444	0.205	0.200	0.217
32			0.832		0.902	0.891	0.866		0.910	0.862	0.717
			0.069		0.430	0.243	0.209		0.236	0.464	0.245
64			0.803				0.871				
			0.075				0.285				

of Fourier modes simulated, see Table 1, first row of data. The ratio R_2 of the time taken by the processors communication and the sum of the times spent for the FFT and for the communication itself is shown in the second rows of data in Table 1.

The ratio R_2 slightly increase on the T3E with the number of processors, and display large fluctuations on the SP3 and ORIGIN (see Table 1, second row), though on average still increasing with the number of processors. The increasing of R_2 with the number of processors is a consequence of the algorithm saturation, while the relevant fluctuations are due mainly to cache effects and an high process to process latency. The large fluctuations are not present on the T3E because the cache size is small compared to the data set and because the operating system interferes with the user process on the computer nodes much less than for the other two architectures.

As an example of the general performances of the present code we show a three-dimensional power spectrum computation of a decaying homogeneous isotropic field at a moderate Reynolds number based on the Taylor microscale ($\mathrm{Re}_\lambda \sim 60$), to which there correspond the set of experimental data by Comte–Bellot and Corrsin [8]. The field is simulated using the direct numerical approach over a mesh of 128^3 points. The spectrum of the turbulent energy is illustrated in Fig. 5 where it is compared both with the experimental spectrum by Comte–Bellot and Corrsin and with the spectrum obtained by Mansour and Wray [17] by means of a direct numerical simulation over 256^3 points. Apart from the region at low wave number that depends on the peculiarities of the initial random field and for which the limitation in prediction due to the finiteness of the computational box is unavoidable, the agreement is very good for $k\eta > 0.02$, i.e. in the region where the energy distribution corresponds to the inertial range.

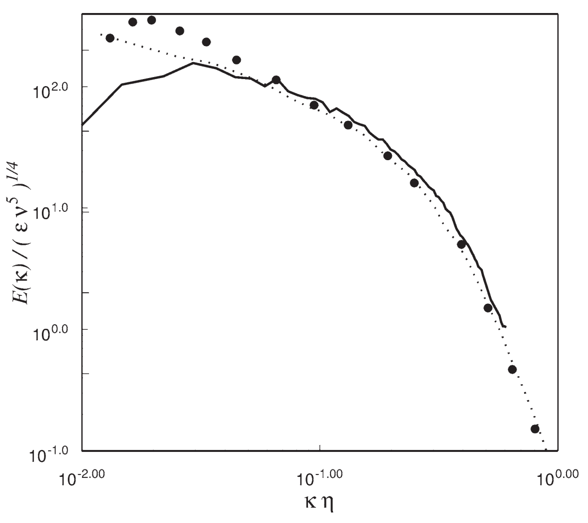

Fig. 5. Energy spectrum of decaying homogeneous and isotropic turbulence: (——) present simulation at $\mathrm{Re}_\lambda = 54$ (Cray T3E), (· · · · ·) 256^3 DNS by Mansour and Wray [17] at $\mathrm{Re}_\lambda = 56$, (●) experimental grid turbulence by Comte–Bellot and Corrsin at $\mathrm{Re}_\lambda = 61$ [8].

5. Conclusions

We have developed a parallel algorithm using the pseudospectral method for the direct numerical simulation of the three-dimensional Navier–Stokes equations solutions. The algorithm is applicable to direct simulations and may be easily modified for use with the large eddy scale simulation method. The novel aspect that this algorithm presents is a data transposition technique directly coupled to the dealiazing procedure based on the rule of the "3/2". Also along the direction of data distribution it allows the use of standard single-processor Fast Fourier Transform (unidimensional and real to real) and keeps at the same time the number of physically significant Fourier modes equal to that corresponding to the domain discretization.

The algorithm has not been conceived for a specific machine. It shows a good portability among different architectures as it is demonstrated by the content of Table 1, where the ratio between the time spent to compute the non linear terms of the Navier–Stokes equations and the total computation time required by one time step is seen to remain nearly constant regardless the machines, the number of parallel processors employed and the spatial discretization adopted. In its turn the fraction of time required by the transposition procedure inside the time interval required by the product computation, i.e. essentially the time spent for the communication among the processors, is very low (few percents) when the behaviour of the machine does not depend on the existence of a cache memory and rises to a 10–20% for the other category of machines, whether working with a number of processors for which the speed-up is still close to the ideal one. For a problem—homogeneous isotropic turbulence—with a mesh ranging from 32^3 to 256^3 the efficiency of the procedure turned out to be very high: inside the interval 0.9–1 for the Cray T3E for n_p up to $N/2$, inside 0.8–1.1 for the IBM SP3 and inside 0.75–1.2 for the SGI ORIGIN 3800 up to $n_p = N/4$. The code was written using the FORTRAN language without optimizations.

Acknowledgements

We acknowledge partial support from CINECA (Centro di calcolo interuniversitario dell'Italia nord-orientale) and from ASI (Agenzia Spaziale Italiana).

References

[1] A.J. Basu, A parallel algorithm for the solution of the three-dimensional Navier–Stokes equations, Parallel Comput. 20 (1994) 1191–1294.
[2] M. Briscolini, A parallel implementation od a 3D pseudospectral bases code on the IBM 9076 scalable POWER parallel system, Parallel Comput. 21 (1995) 1849–1862.
[3] D.A. Briggs, J.H. Ferziger, J.R. Koseff, S.G. Monismith, Entrainment in a shear-free turbulent mixing layer, J. Fluid Mech. 310 (1996) 215–241.
[4] C. Cambon, N.N. Mansour, F.S. Godeferd, Energy transfer in rotating turbulence, J. Fluid Mech. 337 (1997) 303–332.
[5] C. Canuto, M.Y. Hussaini, A. Quarteroni, T.A. Zang, Spectral Methods in Fluid Dynamics, Springer Verlag, Berlin, 1988.
[6] C. Canuto, C. Giberti, Parallelism in a highly accurate algorithm for turbulence simulation, in: D.J. Evans, C. Sutti (Eds.), Parallel Computing. Methods, Algoritms and Applications, Adam Hilger, Bristol and Philadelphia, 1988, pp. 157–168.
[7] S. Chen, X. Shan, High-resolution turbulent simulation using the Connection Machine-2, Comput. in Phys. 6 (1992) 643–646.
[8] G. Comte-Bellot, S. Corrsin, Simple Eulerian time correlation of full and narrow band signals in isotropic turbulence, Fluid Mech. 48 (1971) 273.
[9] K.T. Dang, Evaluation of simple subgrid-scale models for the numerical simulation of homogeneous turbulence, AIAA J. 23 (1985) 221–227.

[10] P.F. Fisher, A.T. Patera, Parallel simulation of viscous incompressible flows, Ann. Rev. Fluid Mech. 26 (1994) 483–527.

[11] Y. Gagne, B. Castaing, A universal representation without global scaling invariance of energy spectra in developed turbulence, C. R. Acad. Sci. Paris, serie II 312 (1991) 441–445.

[12] E. Jackson, Z.S. She, S.A. Orszag, A case study in parallel computing: I. Homogeneous Turbulence on a Hyphercube, J. Sci. Comput. 6 (1991) 27–45.

[13] A. Jameson, H. Schmidt, E. Turkel, Numerical solutions of the Euler equations by finite volume methods using Runge–Kutta stepping schemes, AIAA Paper (1981) 81–1259.

[14] P.K. Jimack, N. Touheed, An Introduction to MPI for Computational Mechanics, University of Leeds.

[15] J. Jimenez, A.A. Wray, P.G. Saffman, The structure of intense vorticity in isotropic turbulence, J. Fluid Mech. 255 (1993) 65–90.

[16] M. Lesieur, O. Métais, New trends in large eddy simulations of turbulence, Ann. Rev. Fluid Mech. 28 (1996) 45–83.

[17] N.N. Mansour, A.A. Wray, Decay of isotropic turbulence at low Reynolds number, Phys. of Fluids 6 (1996) 808–813.

[18] C. Menevaeau, J. Katz, Scale-invariance and turbulence models for large-eddy simulation, Ann. Rev. Fluid Mech. 32 (2000) 1–32.

[19] Message Passing Interface Forum, MPI: a Message-Passing Interface Standard, 12 June 1995.

[20] S.A. Orzag, Numerical simulation of incompressible flows within simple boundaries, I. Garlekin (spectral) representation, Stud. Appl. Math. 50 (1971) 293–327.

[21] R.B. Pelz, The parallel Fourier pseudospectral method, J. Comput. Phys. 92 (1991) 296–312.

[22] T.P. Ray, Jets: a star formation perspective, in: S. Massaglia, G. Bodo (Eds.), Astrophysical Jets: Open Problems, Gordon & Breach Sience Publishers, 1996, p. 173.

[23] R.S. Rogallo, ILLIAC program for the numerical simulation of homogeneous incompressible turbulence, Nasa TM-73203, 1977.

[24] R.S. Rogallo, Numerical experiments in homogeneous turbulence, Nasa TM-81315, 1981.